GAG News

GAG News 175

Ausgabe 175 Mai/Juni 2021

C-Programmierung

Fenster die Rollen und viel Text enthalten

PHP - Lua - Perl - Python

Systempflege und Libraries

Wakefield Show und weitere Neuigkeiten... und Tipps

Die deutsche Zeitschrift für RISC OS-Computer

RISC OS

C-Programmierung

Praxis

Software

Shows

Kurz Notiert

GAG

Rückblick

An sich sollte man denken, in so einer mehr oder weniger runden Nummer ist ein Rückblick – aber den hebe ich mir für die Ausgabe zum 30-jährigen Jubiläum auf, sprich Ausgabe 180, denn da wird soetwas eh erwartet und zwei Rückblicke in so kurzer Folge wären dann doch ein wenig viel.

Über Input von Dir für den Rückblick in News 180 – ob fein ausformuliert oder in Stichworten – würde ich mich sehr freuen, zumal ich sicher bin, dass ich das eine oder andere Highlight eventuell übersehen werde. Danke!

GAG-Treffen

Ein Clubtreffen im realen Leben wird es dieses Jahr eventuell nicht geben, da sich die Corona-Zahlen leider zu drastisch in die falsche Richtung entwickeln. Ob wir ein Online-Treffen auf die Beine stellen können, ist fraglich, da nicht jeder die entsprechende Technik zur Hand hat, da die Onlinekonferenzsysteme unter RISC OS eher nicht nutzbar sind (die eventuell geschürte Hoffnung auf dafür geeignete Software war nicht mehr oder weniger als die jährliche Ente unter dem Deckmantel des Aprilscherzes).

April, April

Vielleicht hast Du den Aprilscherz in der letzten News gefunden… Er war auf Seite 27 unten in der Mitte: Dass jemand GoToMeeting für RISC OS anpasst, ist bestenfalls Wunschdenken.

Viel Spaß beim Lesen!

Never change a running System

Updates oder auch nicht

Herbert zur Nedden

In der EDV gibt es eine bekannte Regel: „*Never change a running System*“, sprich wenn’s funktioniert, ändere es lieber nicht.

Der Umkehrschluss hieraus ist übrigens, dass man, wenn etwas urplötzlich nicht mehr funktioniert, als erstes schaut, was geändert wurde, da die Änderung ein guter Kandidat für das Nichtfunktionieren ist.

Leider ist eben dieser Ansatz heutzutage ein Luxus, den man sich eigentlich nicht mehr leisten kann – und das nicht etwa, weil man immer neue oder leistungsfähigere Dinge haben will, sondern weil fast keine Software fehlerfrei ist und einige der Fehler ebenso, wie Designschwächen ausgenutzt werden, um Schaden anzurichten…

… außer unter RISC OS!

Sicherheit

Der Klassiker heutzutage in dieser Kategorie sind die berühmt-berüchtigten Securityfixes bei denen es außer Frage steht, ob man sie einspielt – lediglich bei Rechnern, die keinerlei Kontakt zur Außenwelt haben, kann man sich überlegen, ob. Dabei bitte „keinerlei Kontakt“ nicht mit „keine Anbindung ans Internet“ verwechseln, da Schadsoftware auch per USB-Stick verbreitet werden kann.

Selbst die geprüften Originalmedien von Softwareherstellern können verseucht sein – ich erinnere mich gut daran, dass es sogar mal von einem RISC OS-Softwareanbieter eine CD gab, die einen Virus drauf hatte. Ernsthaft im doppelten Sinne! Früher gab es für RISC OS Schadsoftware und eine davon war auf der CD – einige Veteranen erinnern sicherlich auch noch Virenschutztools für RISC OS.

Eine Nebenwirkung hiervon ist auch der „Zwang“ zu gelegentlichen Upgrades, da Hersteller verständlicherweise ihre allzu alten Softwareversionen nicht mehr pflegen und auch bei den unterstützten Betriebssystemversionen eine gewisse Aktualität erwarten. In Firmen mit vielen Servern ein Quell der Freude, da es nicht immer möglich ist, das Betriebssystem in-place upzugraden und zum Teil die eingesetzte Software das neue OS nicht mag. In dieser Hinsicht haben unixoide Systeme nach meiner Erfahrung schon die Nase vorne, da bei denen das Upgraden tendenziell einfacher und sauberer klappt, als bei Windows.

In diesem Zusammenhang spielt auch mit, dass nach meinem Eindruck Software für unixoide Systeme eher die Regeln des Betriebssystems einhalten, als die unter Windows, was schon die Frage betrifft, wohin kommt der Code, wohin die Einstellungen und wohin die Daten. So scheint es zum guten Ton zu gehören, dass die Installationsroutinen für Windowsapplikation extra Rechte im System vergeben oder gar verlangen, dass man einen Virenschutz während der Installation abschaltet.

Da haben wir mit RISC OS echt gute Karten, denn an Securityfixes für RISC OS kann ich mich nicht so wirklich erinnern. Und das Gros der Software läuft auch unter älteren RISC OS-Versionen und das Upgraden von RISC OS selbst ist auch eher einfach. Allerdings gibt es zwischenzeitlich unter RISC OS durchaus auch einige Softwaren, die RISC OS 5 voraussetzen oder gar halbwegs aktuelle Prozessoren.

Virenschutz

Ein Rechner ohne aktuellen Virenschutz – guter Witz heutzutage. Mehr als grenzwertig gewissermaßen ist meiner Meinung nach, dass auf neuen Rechnern häufig ein kommerzieller Virenschutz mit Dreimonatslizenz vorinstalliert ist, denn nach Ende der drei Monate läuft der Virenschutz weiter, so dass der eher unbedarfte User meint, alles gut (und ein Upgrade auf eine neue Version brauche ich gerade nicht), obwohl er sich keine neuen Patterns holt, und somit schnell recht wirkungslos ist.

Dass ein Virenschutz nur eine gewisse Sicherheit bietet, da es einige sehr fähige Programmierer gibt, die Schadsoftware programmieren und das so, dass die handelsüblichen Virensoftwaren sie nicht erkennen, bis der Hersteller des Virenschutzes das neue Kleinod kennen lernt und per Patternupdate nachrüstet, ist sicher hinlänglich bekannt.

In dem Zusammenhang schon erstaunlich ist, dass gerade Smartphones, die bekanntlich nicht einmal einen Router als erste Firewall vor dem Internet haben, häufig keinen Virenschutz haben. Aber immerhin ist hier ein wenig mehr Sicherheit, als auf dem normalen PC dadurch gegeben, dass man Software für die mobilen Helfer i. a. nur aus dem passenden Appstore installieren kann und das, was dort angeboten wird, einer gewissen Kontrolle unterliegt – doch nur einer „gewissen“, wie die Vergangenheit gezeigt hat. Das andere, was hier hilft ist, dass das Betriebssystem von sich aus dem Treiben von Apps einige Einschränkungen auferlegt.

Für RISC OS gab es eine Zeit lang übrigens auch ein paar Viren- und Malwareschutzprogramme und das Modul VProtect, dass zu einem dieser Tools gehörte, hat erstaunlich lange in RISC OS unverändert (!) überlebt. Aber die Zeiten sind vorbei.

Rechtebeschränkung

Eine weitere Möglichkeit für Sicherheit zu sorgen ist, dass man als Benutzer auf seinem eigenen Rechner nicht alles tun darf – klingt erst mal total bescheuert, aber…

Bei der normalen Nutzung meines Rechners brauche ich nur wenig Rechte, da es im Kern genügt, dass ich einige Programm starten und mit meinen Daten jonglieren darf. Neue Software zu installieren und dabei im Betriebssystem zu verankern oder gar Systemkomponenten und Treiber zu ändern, ist nur selten notwendig.

Mit eingeschränkten Rechten arbeiten zu können schützt somit das Betriebssystem und damit den Rechner durchaus, weil dann auch versehentlich von mir gestartete Schadsoftware ebenso wirkungsbeschränkt ist. Dabei kann man das je nach Betriebssystem auch so weit treiben, dass nur bestimmte Programme gestartet werden dürfen und einige sogar in Bezug auf die Verzeichnisse, auf die sie schreiben dürfen, limitiert sind bis dahin, dass zunehmend für Browser sog. Sandboxes zum Einsatz kommen, die diesen noch stärker vom Rest des Systems abschotten.

Linux macht es seit eh und je, da da der normale Useraccount kein Administrator ist, sprich man für die Installation vor Software etc. erst den Account wechseln muss, was aber recht einfach ist. Und es gibt für Linux auch Tools, die das System so, wie im vorherigen Absatz angedeutet, weiter härten. Obendrein kann man durch sog. chrooten Dienste und Anwendungen unter Linux noch weiter einsperren und somit vom Kern des Systems fernhalten.

Microsoft hat einiges davon in Windows nach und nach umgesetzt – genaugenommen ist es an sich schon recht lange möglich, unter Windows für die Administration und die Nutzung separate Accounts zu nutzen, aber das wurde eher nicht gemacht, weil zu umständlich und weil diverse Softwarehersteller es dem geneigten Benutzer gründlich erschwerten (ergo: verleideten), da sie die Regeln, was wo zu speichern ist, mehr oder weniger konsequent ignoriert haben.

Microsoft hat einiges davon in Windows nach und nach forciert – ist aber mit Blick auf die Userschar Kompromisse eingegangen, die das Wechseln in den administrativen Modus nicht allzu schwer machen, indem sie einem sogar den Wechsel des Accounts ersparen und ein Klick genügt, um volle Rechte zu haben. Dazu kommt, dass auch viele Softwarehersteller sich (immer noch) nicht an die Regeln halten und für ihre Software unnötige Schreibrechte brauchen. Daher hat Microsoft sogar ein wenig getrickst, so dass Software scheinbar in ihr Programmverzeichnis schreiben darf, die Dateien landen aber in einem Verzeichnis im Userprofil, dass virtuell über das Programmverzeichnis gelegt wird (also etwas wie LayerFS). Das funktioniert je nach Anwendung mehr oder weniger gut, da beim Löschen eigener Dateien das Original aus dem Programmverzeichnis wieder erscheint, was nicht immer passen tut.

Mit Windows 10 bietet der Windows Defender auch die Option, Verzeichnisse so zu schützen, dass nur bestimmte Programme in diese schreiben dürfen – an sich eine gute Idee, weil das Malware zusätzlich behindern kann, aber auf der anderen Seite auch ein wenig lästig ist, wenn man nicht mehr aus Word heraus überall auf der Platte speichern darf, ohne diese Einschränkungen hie und da zu lockern. Aber es ist ja bekannt, dass Sicherheit tendenziell mit einem gewissen Komfortverlust einhergeht. Es gilt, einen guten Mittelweg zu finden, denn Sinn macht dieser Schutz schon.

Da sind wir unter RISC OS natürlich viel, viel bequemer dran, da RISC OS so gar keinen Schutz bietet. Ok, fairerweise sollte ich erwähnen, dass man Dateien schreibschützen kann und es auch unterschiedliche Rechte für lokale Zugriffe und die übers ShareFS-Netzwerk gibt, aber das ist kein wirklicher Schutz, da man zumindest bei lokalen Zugriffen den Schreibschutz einfach entfernen kann.

Multiuser

Bitte diese Rechtelimitierung nicht mit dem Ansatz, auf einem Rechner mehrere Benutzer anlegen zu können, verwechseln, was RISCOS Ltd mit Select ja mal eingebaut hat. Das mit mehreren Benutzern dient dafür, dass jeder User seine eigenen Einstellungen hat, sprich es gab nicht ein !Boot.Choices, sondern eines pro User (ohne jedweden Schutz). Und dann haben sie auch noch Hardwareprofile vorgesehen, aber die waren i. a. deaktiviert. Was in Select übrigens, anders als beim Mainstream hierbei nicht vorgesehen wurde, sind unterschiedliche Datenbereiche, aber das liegt wohl daran, das es keinen Standardpfad zu den Daten eines Users gibt, wie das Homeverzeichnis beim Mainstream.

Auf Mainstream-Betriebssystemen sind die Einstellungen und Datenbereiche pro Anwender vor dem Zugriff durch die anderen User des Rechners in der Regel geschützt – aber natürlich nicht vor denen mit Administratorrechten. Und mit ein wenig Mühe kann man sogar den Administrator aussperren.

Aber auf einem privaten PC ist das heutzutage tendenziell weniger interessant – es war früher hingegen schon sehr hilfreich, als PCs noch richtig teuer waren, so dass sich mehrere Familienmitglieder einen PC geteilt haben, was heute eher selten ist.

Zwischenbilanz

RISC OS hat schon ein Alleinstellungsmerkmal, da es den User nicht zwingt, aus Sicherheitsgründen irgendwelche Updates oder Upgrades einzuspielen – somit sind wir auf diesem OS durchaus recht gut in der Lage das „Never change a running System“-Dogma zu befolgen.

Allerdings sollten wir schon, wenn eine neue, stabile RISC OS-Version von RISC OS Open oder bei R-Comp-Ware von denen herausgegeben wird, diese mal einspielen – den einen oder andere Bug so loszuwerden, kann nie schaden.

Und seit das Thema TLS auch unter RISC OS etwas mehr beachtet und betreut wird, was sich primär in Form des Moduls AcornSSL zeigt, kommen wir langsam in den Genuss von einer Art Securityupdates. Und ich befürchte, dass mit Iris ein weiterer Kandidat für gelegentliche Securityupdates auf uns zukommt .

RISC OS hat noch eine Eigenschaft, die einem das Leben einfach macht: im Prinzip keine Security, so dass ich schalten und walten kann, wie ich will.

Neues

Natürlich gibt es einen richtig guten Grund, doch etwas zu ändern: Neue Software oder neue Versionen von Software, die Features mitbringen, die man gerne haben möchte.

Gemein ist, dass just dieser Fall auch die „Nebenwirkung“ haben kann, dass man zuvor das Betriebssystem auf einen neueren Stand bringen muss – ob das nun gar eine neue Version von RISC OS ist, oder nur das Einspielen von Modulen oder Libraries, sei dahingestellt. Auf der anderen Seite kann man auch nicht ernsthaft von Entwicklern erwarten, Betriebssystemversionen zu supporten, die allzu historisch sind.

Unter RISC OS verlangen einige Softwaren mindestens RISC OS 4; andere RISC OS 5 bzw. sogar eine Mindestversion davon, weil sie Features davon benötigen. Das halte ich auch für legitim, denn warum soll ich Dinge, die das OS mittlerweile bietet, in meinen Programmen nachprogrammieren?

Es gibt, wie einige sicherlich wissen, auch RISC OS 6, bei dem es sich im Kern um RISCOS Ltd's 32bittiges RISC OS 4 für den A9home handelt – das ist somit älter als das aktuelle RISC OS 5.

Ergo kann es, wenn auch mit überschaubarer Wahrscheinlichkeit, passieren, dass eine neue oder erneuerte

Software nur nutzbar ist, wenn Du ein aktuell genügendes RISC OS nutzt – dabei sei angemerkt, dass außer für den A9home RISC OS 5 für alle halbwegs aktuellen Rechner verfügbar ist (ja, das „aktuell“ ist gerade sehr großzügig ausgelegt).

Auch gibt es die eine oder andere Software, die bestimmte Hardware benötigt oder empfiehlt – mal ob der Prozessorfeatures (z. B. VFP für die Performance) und mal ob verfügbarer Komponenten des Rechners (z. B. GPIO beim Raspberry Pi).

Support

Und dann gibt es noch einen weiteren Grund, Updates oder Upgrades einzuspielen: Der Hersteller hat eine neue Version auf den Markt gebracht und verkauft diese als Upgrade und Du kaufst die einfach, um eben den Hersteller so (finanziell) zu supporten und zu motivieren, weiterzumachen (den Fall, dass Du upgradest, um in den Genuss neuer Features zu kommen, habe ich im vorherigen Abschnitt schon betrachtet).

Das ist ein durchaus legitimer Ansatz, bei dem allerdings im Laufe der Jahre einige Hersteller mit dem Wohlwollen der Kunden nach meinem Geschmack etwas arglos umgegangen sind. Legendär war RISCOS Ltd, die mit klaren Zusagen starteten, welche Features sie zu wann in RISC OS Select einbauen wollen und wie oft man neue RISC OS-Versionen zum Ausprobieren bekommen würde – und beides ohne den Preis anzupassen, nicht einhielten. Erstaunlich, wie viele Kunden da wie lange mitgespielt haben.

Ich habe lange von Software, die ich regelmäßig nutzte, kein Upgrade ausgelassen, um zum einen die funktionalen Neuerungen zu bekommen aber auch, um die Hersteller so zu supporten – und da kamen schon erkleckliche Sümmchen zusammen.

Aber bei einigen verschob sich der Nutzen zunehmend auf „Support des Anbieters“, weil sie entweder fast nix neues eingebaut haben oder nur Dinge, die ich so gar nicht brauchte. Mal ein Upgrade, dass mir funktional eher wenig nützt ist ja ok, denn andere User haben andere Bedürfnisse, aber leider passierte es zunehmend, dass bei neuen Versionen die Relation zwischen den neuen Funktionen (unabhängig von meinem Nutzen) und dem Preis in eine für meinen Geschmack zu heftige Schieflage geriet.

Seit einiger Zeit mache ich es daher so, dass ich nur noch die Upgrades kaufe, die mir irgendeinen Nutzen bringen (oder zumindest versprechen, das zu tun) – und das Gros der betroffenen Hersteller weiß, welche Features ich reizvoll finden würde… und das schon länger. Dieser Schritt ist mir nicht leichtgefallen, weil mir schon klar ist, dass die Hersteller im RISC OS-Umfeld nicht gerade viele Kunden haben, aber es gibt einfach Grenzen – da kann ich das Geld besser in Bounties von RISC OS Open oder einem Restaurantbesuch mit meiner Frau anlegen. Und wenn es dann mal eine neue Version von was-auch-immer gibt, die reizvoll ist, dann lange ich halt dann zu – auch, wenn dann das eine Upgrade mehr kostet, weil ich Zwischenschritte ausgelassen habe.

Neugier

Und es gibt natürlich noch einen Grund dafür, etwas zu ändern: Die pure Neugier! So probierte ich früher eigentlich immer wieder mal neue RISC OS-Beta-Versionen aus (aktuell hapert es daran, dass es kein bei RISC OS Open zu saugendes Image für den mini.m geeignet ist, weil R-Comp nicht alles notwendige an RISC OS Open gegeben hat) oder das aktuelle !NetSurf.

Aktuell

Bei Updates aber auch Securityfixes ist es nicht unüblich, dass diese in Firmen erst auf ein paar ausgewählten Systemen eingespielt werden, da es vereinzelt vorkommt, dass sie unerwünschte Nebenwirkungen haben und man die nicht gleich in der Produktion erleben möchte. Für den privaten Rechner „simuliert“ man das, indem man bei Updates ggf. einfach mal ein wenig wartet (bei Securityfixes nur kurz, bitte) und in Mailinglisten oder Foren mitliest, ob andere von Problemen berichten.

Das gilt übrigens auch für RISC OS, wobei man natürlich eher risikoarm ein neuen !NetSurf probieren kann (den alten vorher in old!NetSurf umbenennen), da der Weg zurück ruckzuck zu schaffen ist; bei RISC OS selbst, hängt es ein wenig von der Hardware ab. Beim mini.m, wo die SD-Karte nur das ROM-Image vorhält, da alles andere auf der SATA-SSD liegt, genügt eine zweite SD-Karte für gefahrlose Experimente. Anderenfalls warte einfach einen Moment, da es genug „early adopters“ gibt, die die Bugs für Dich finden werden :-)

Damit fährt man in der Regel recht gut, aber halt nicht immer, wie sie Anfang März mal wieder gezeigt hat, da Unmengen von Exchange-Servern infiziert wurden, da der verfügbare Fix nicht schnell genug eingespielt wurde. Und dass die Hersteller ihre Kunden aktiv informieren und Sicherheitsforen ebenfalls, impliziert, dass die zuständigen Techniker es hätten wissen können. Ist übrigens nicht das erste Mal, wo das Trägheitsmoment beim Einspielen von Securityfixes als Folge hatte, dass viele IT-Systeme gnadenlos und erfolgreich infiziert wurden.

Security by Obscurity

Insgesamt gilt bei RISC OS gewissermaßen, dass wir recht problemlos das *„Never change a running System“* umsetzen können, da der Hauptgrund, der dagegen spricht, die Sicherheit, dank *„Security by Obscurity“* automatisch gewährleistet ist – wobei das „Obscurity“ bei RISC OS, dass ja Quelloffen ist, weniger im Sinne von „Undurchsichtigkeit“, sondern „Obskur“ zu interpretieren wäre. Selbst, wenn es einige neue User für RISC OS dank der Bemühungen von RISC OS Open, RISC OS Developments, Cloverleaf und dem Raspberry Pi als preiswerter Einstiegsdroge oder warum sonst auch immer geben dürfte, wird es sicherlich noch etwas dauern, bis die „kritische Masse“ erreicht wurde, damit es sich für die Malwarehersteller lohnt, RISC OS zu „supporten“.

Fazit

Insgesamt schon genial, ein OS zu haben, wo man ohne irgendwelche Hindernisse schalten und walten kann und dass trotzdem recht sicher ist und obendrein gut auf einer Hardware nutzbar ist, die man für unter 50 € bekommt. Dass es für viele Dinge dank passender Software hervorragend geeignet ist, hilft natürlich ungemein.

Es hat also echte Vorteile, so ein schönes, exotisches Betriebssystem zu haben. ❍

Fenster mit Scrollbalken

C-Programmierung unter RISC OS #13

Herbert zur Nedden

Nummer 13 gibt es auch – ich hoffe, dass ist für Dich kein schlechtes Omen!

In der letzten Ausgabe der News zauberten wir erstmals ein Fenster auf den Schirm und wussten es auch wieder zu schließen. Bevor wir ein wenig Inhalt abgesehen von einzelnen Icons spendieren, lassen wir das Fenster wachsen, schrumpfen sowie per Rollbalken den sichtbaren Ausschnitt desselben durch den User wählen, damit das mit dem Inhalt etwas mehr Raum bekommt.

Rollenspiele

So, nun bekommt das Fenster gleich die „volle Dröhnung“, sprich all die netten Symbole zum Ändern der Größe und zum Scrollen kommen auf den Rand. Die Window-Flags werden somit zu

```
1111 1111 0000 0000 0000 0101 0000:
#define WindowFlags_AllToolsWindow
    0xFF000052
```

Zur Erinnerung und damit Du nicht nachschlagen musst, die obersten acht Bit im Flag, die wir nun alle gesetzt haben:

24 Fenster hat ein „Pack-nach-Hinten-Symbol“
25 Fenster hat ein Schließen-Symbol
26 Fenster hat einen Titelbalken
27 Fenster hat ein Größenumschalt-Symbol
28 Fenster hat einen vertikalen Rollbalken
29 Fenster hat ein Größenanpassungs-Symbol
30 Fenster hat den horizontalen Rollbalken
31 Bit 24-30 werden beachtet (statt der alten Bits 0, 2, 3 und 7)

Ausschnitt

Damit das mit den Balken zum Rollen einen Sinn hat, gibt es eine neue Funktion, die ein Fenster anlegt, aber nicht die ganze Fläche sichtbar macht, sondern nur einen Ausschnitt:

```
// Create a big window
t_Window *CreateBigWindow(char *Title, unsigned Flags, int Icons,
                          int MaxX, int MaxY, int VisX, int VisY) {
    t_WindowHeader *WindowHeader;
    WindowHeader = (t_WindowHeader*)CreateBasicWindow(Title,Flags,Icons,MaxX,MaxY);
    // Just set visible size - we move to center later
    WindowHeader->Visible.MaxX = VisX;
    WindowHeader->Visible.MaxY = VisY;
    // We want top left but bottom right is (0,0) so we scroll up
    WindowHeader->ScrollX = 0;
    WindowHeader->ScrollY = MaxY;

    return((t_Window*)WindowHeader);
}
```

Der Aufruf ist dem alten CreateBasicWindow natürlich ähnlich – er hat nur zwei Parameter mehr, in denen die sichtbare Fläche angegeben wird.

Die neue Funktion CreateBigWindow nutzt die alte für die Basis und ändert dann lediglich die Größe des sichtbaren Bereichs auf die neuen Limits und damit wir in der linken, oberen Ecke starten, wird das Fenster noch nach oben gescrollt, indem ScrollY entsprechend gesetzt wird, da die Koordinate (0,0) die linke untere Ecke ist.

Nomenklatur

Im realen Leben kennen wir Fenster – das sind die Löcher in der Wand (hierzulande meist mit einer Glasscheibe versehen), durch die man nicht nur hinaus, sondern auch hineinschauen kann und dabei sieht man einen Teil des Inneren des Hauses.

Auf einem Rechner mit Fensteroberfläche ist es ähnlich, da ein Fenster einen Blick auf alles oder einen Ausschnitt dessen bietet, was da Programm da dem geneigten User präsentiert.

Soweit noch total einfach und einleuchtend – doch nun komme ich zu den Begrifflichkeiten, die RISC OS in diesem Fall für uns hat:

Für SWIs (und die nutzen wir bei der Programmierung) ist „Window“ nicht nur das Fenster auf dem Monitor, sondern auch das Hintergründige, das das Programm absondert. So kennt es zum einen die sog. „Visible Area“, bei der es sich um das handelt, was wir als Fenster auf dem Monitor sehen: Es ist die rechteckige Fläche, die den Fensterinhalt ausmacht – und das natürlich auch, wenn sie nicht ganz zu sehen ist, weil über den Monitorrand hinausgeschoben oder (teilweise) durch andere Fenster überdeckt. Zum anderen gehört zu einem „Window“ auch das, worauf das Fenster einen Blick gewährt, „Work Area“ genannt: Es ist das Rechteck, in das das Programm das packt, was „durch“ das Fenster zu sehen ist – ob nun in Gänze oder nur ein Ausschnitt davon.

Bevor ich diese Angaben in Relation zueinander setze und erläutere, hier zur Erinnerung der WindowHeader:

```
typedef struct
    t_BoxenLuder   Visible;
    int            ScrollX, ScrollY;
    t_WindowHandle WindowStack;
    int            WindowFlags;
    char           TitleForground;
    char           TitleBackground;
    char           WorkareaForground;
    char           WorkareaBackground;
    char           ScrollbarBackground;
    char           ScrollbarForeground;
    char           TitleFocusBackground;
    char           ExtraFlags;
    t_BoxenLuder   Workarea;
    t_IconFlags    TitleIconFlags;
    int            WorkareaButtonType;
    t_SpriteArea  *ToolSpriteArea;
    int            MinArea;
    t_IconData     TitleIconData;
    int            IconCount
} t_WindowHeader;
```

Der Datentyp t_Boxenluder ist, wie Du sicher erinnerst, nichts anderes als eine Struktur mit vier int-Werten namens MinX, MinY, MaxX, MaxY, sprich beschreibt ein Rechteck (vlugo eine Box).

Die eigentliche Fläche, die das Programm mit Informationen füllen darf, ist die Workarea, wobei es überra-

schend ist, dass die Fläche nicht nur durch Breite und Höhe festgelegt wird und z. B. die linke untere Ecke (0,0) ist, sondern frei wählbar.

Ob sie damit den Programmierern entgegenkommen wollten, für die der Ursprung eines Fensters nicht die linke untere, sondern die linke obere Ecke ist? Das halte ich für wahrscheinlich, zumal viele Programmierbeispiele genau den Ansatz verfolgen. In dem Fall muss man halt nur mit negativen Koordinaten in der vertikalen operieren, sprich die Workarea ist dann von (0,0) bis (Breite,-Höhe). Und weiter unten wird sogar deutlich, warum genau das viel Sinn macht.

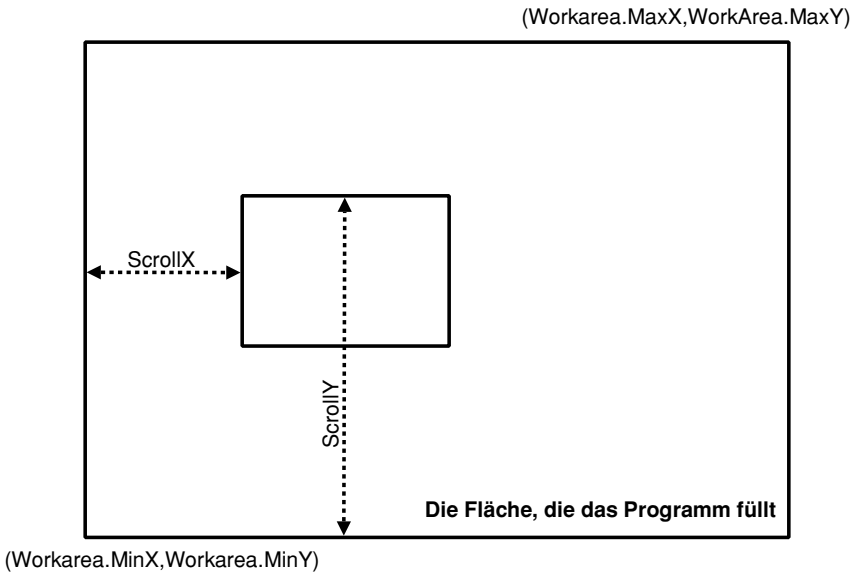
Auch bei Icons kann man dieses Spiel treiben, da die in einer Bounding Box leben und auch die kann ich ebenso, wie die Workarea, so definieren, dass (0,0) und damit deren Ursprung links oben ist. Bei Icons ist der Ursprung insofern wichtig, da deren Positionierung eben diesen nutzt.

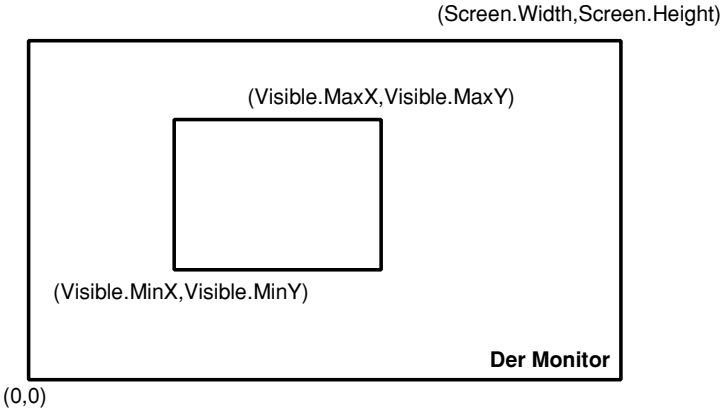
Es ist in diesem Zusammenhang ratsam, die Workarea der Fenster und die Bounding Boxes von Icons einheitlich zu definieren – zumindest aus Sicht des Programms –, denn sonst kann es unübersichtlich werden.

Die Fläche, die ein Fenster auf dem Monitor belegt, wird durch die Koordinaten von Visible umrissen und dabei handelt es sich um Bildschirmkoordinaten, so dass hier natürlich die volle Flexibilität von (MinX,MinY) bis (MaxX,MaxY) notwendig ist, damit das Fenster nicht zwingend unten links in der Ecke spawnt.

Die dritte wichtige Angabe in diesem Zusammenhang ist der Scrolloffset, der in ScrollX und ScrollY abgelegt ist. Damit wird angegeben, wo sich das Fenster (Visible) über der WorkArea befindet und damit, welcher Ausschnitt der Workarea im Fenster sichtbar sein soll.

Anmerkung: Meine Routine, die den WindowHeader initial füllt, setzt die Visible-Fläche initial von (0,0) bis (Breite,Höhe), womit das Fenster links unten auf dem Schirm erscheinen würde, aber vor dem Öffnen des Fensters werden diese Werte dann so angepasst, dass dieses Rechteck beim Öffnen auf dem Schirm mittig erscheint (die Funktion MoveToCenter macht das).





Symbolisch

Damit wir auch was zu sehen haben, spendiere ich meinem Fenster mal in Eckennähe ein paar Texticons. Sie bekommen einen weißen Hintergrund, damit gut zu sehen und als Textinhalt die Positionsanagabe von links oben (LO) bis rechts unten (RU).

Bei der Positionierung der Icons muss ich ein wenig rechnen. Links ist einfach, da ich da weiterhin den Abstand von 20 nutze. Für ein Icon rechts muss ich von der Breite der Fensterarbeitsfläche nicht nur den Abstand von 20, sondern natürlich auch die Breite des Icons (also die 260) abziehen. In der Vertikalen ist es analog – unten genügt der 20er-Abstand und für oben ziehe ich von der Arbeitsflächenhöhe die Iconhöhe von 44 und die bekannten 20 ab.

Damit die Chose im Code ein wenig transparenter ist, ist die Arbeitsfläche 1200 breit aber „nur“ 1000 hoch, denn so sind die Ausdehnungen in der Horizon- und Vertikalen gut voneinander unterscheidbar.

```
// A bigger window with scroll bars
// Note that the window bottom left corner is (0,0); the top right one is (1200,1000)
// The icons are 44 OS units high (the usual value for text icons) and 260 units wide
MyMainWindow = CreateBigWindow("Big
    Window",WindowFlags_AllToolsWindow,4,1200,1000,200,100);
FillTextIcon(&(MyMainWindow->WindowIcon[0]),"LO",True,20,1000-44-20,260,44);
FillTextIcon(&(MyMainWindow->WindowIcon[1]),"LU",True,20,20,260,44);
FillTextIcon(&(MyMainWindow->WindowIcon[2]),"RO",True,1200-260-20,1000-44-20,260,44);
FillTextIcon(&(MyMainWindow->WindowIcon[3]),"RU",True,1200-260-20,20,260,44);
```

Programm

Man nehme also das alte WinTwo vom vorherigen Mal, ersetze den Aufruf von CreateBasicWindow samt dem Betanken des einen Icons durch obigen Aufruf von CreateBigWindow und dem Füllen der nunmehr vier Icons und das war's auch schon – und heißen tut das neue programm UpAndDown.

OK, ich habe das Symbolleisten-Menü ein wenig eingekürzt auf nur noch die Klassiker „Info" und „Quit", die mit „Ego" und „Tschüß" beschriftet sind.

Das Fenster kann ich munter scrollen und durch die Gegend schieben. War doch nicht so schwer, oder?

Redraw

Wenn im Fenster nur Objekte enthalten sind, die das Wimp eigenständig ausgeben kann, sind wir fein raus und lassen das Wimp die Arbeit alleine machen. Aber es gibt auch Fenster, wo das Programm dem Wimp dabei unter die Arme greifen darf und das wollen wir mal in Augenschein nehmen, da ich im nächsten Schritt ein wenig Text in das Fenster zaubern will – mal mit und mal ohne einem Text-Icon pro Textzeile.

Das Redraw-Event kommt zum Zuge, wenn mein Fenster oder Teile davon „plötzlich" sichtbar werden; beim reinen Verschieben hat das Wimp ja schon alle Informationen und kopiert einfach den Bildschirminhalt von der alten an die neue Position.

Und da Fensterinhalte durchaus recht aufwändig zu erzeugen sein können – man denke nur an ArtWorks-Grafiken – zerlegt das Wimp die sichtbare Fensterfläche in Rechtecke und bittet die Applikation diese Flächen jeweils zu füllen, sprich just den Teil des Fensterinhalts, der dem jeweiligen Rechteck entspricht, zu liefern. Und da das Wimp beim Verschieben von Fenstern die Teile, die schon sichtbar sind, selbst verschiebt, fordert es das Neuzeichnen nur für die Teile der Fläche vom Programm an, die vorher unsichtbar waren. Beim Vergrößern und Scrollen von Fenstern gilt das entsprechend.

Und sind wir faul, können wir dem Wimp einfach immer den kompletten Fensterinhalt liefern – das Wimp wird's dann schon richten, sprich nur das sichtbare zur Anzeige bringen.

Code

Also verklickern wir dem Wimp, erst einmal, dass wir beim Neuzeichnen des Fensters involviert werden wollen, wozu wir Bit 4 in den Window-Flags löschen:

```
1111 1111 0000 0000 0000 0100 0000:
#define
    WindowFlags_AllToolsWindowRedraw
    0xFF000042
```

Doch nun müssen wir dann auch das tun, was wir dem Wimp zugesagt haben und auf den Redraw-Request qualifiziert reagieren.

Das Wimp meldet sich bei Bedarf beim Wimp_Poll mit dem Window-Redraw-Request und kaum ist der eingetrudelt, dürfen wir ihn in einer Schleife bedienen, da das Wimp unser Fenster nicht am Stück anfordert, sondern die neu zu füllende Fläche in sich nicht überdeckende Rechtecke zerteilt und eben diese nach und nach gefüllt haben möchte. Grund für diese Stückelung ist, wie schon angedeutet, dass es Applikationen gibt, die für das Ausgeben der Fensterinhalte einiges an Aufwand treiben dürfen und da kann es der Performance sehr zugute kommen, wenn die wirklich nur die sichtbaren Bereiche neu erzeugen und zur Anzeige bringen.

Nach Erhalt des Windows-Redraw-Requests rufen wir erst den SWI Wimp_RedrawWindow auf und dann in einer Schleife Wimp_Get Rectangle bis zum bitteren Ende.

Dabei ist das Wimp so nett, und ruf den ersten Wimp_GetRectangle automatisch auf, sprich das Ergebnis vom SWI Wimp_RedrawWindow ist direkt das, was der erste Wimp_GetRectangle liefert. In den Handbüchern wird das als „der SWI Wimp_RedrawWindows kommt via Wimp_GetRectangle" zurück formuliert. Ich finde diese Vorgehensweise leicht verwirrend, aber immerhin wird so ein SWI-Aufruf gespart, was sicherlich zu Urzeiten, wo die ARM-Prozessoren eher schwachbrüstig waren ein Quäntchen Performance gebracht hat.

Der Wimp_GetRectangle (und der Wimp_RedrawWindow somit auch) liefert und die Information, welcher Teil des Fensters dran ist, betankt zu werden:

```
// Block for Wimp_GetRectangle
typedef struct {
    t_WindowHandle WindowHandle;
    t_BoxenLuder   Visible;
    int            ScrollX, ScrollY;
    t_BoxenLuder   Rectangle;
} t_WimpGetRectangle;
```

Das Neue, was wir hier bekommen ist das Rechteck „Rectangle", dass neu zu zeichnen ist. Was wir übrigens hier nicht bekommen ist die WorkArea – aber die kennen wir, da wir das Fenster angelegt habe, aber praktisch wäre es schon gewesen.

Nun brauchen wir auch dies in Wimpy.h:

```
#define ReasonCode_RedrawWindow 1
```

Und dann, da unser Fenster ja an sich noch ohne unser Dazutun korrekt erscheint, nur folgende „komplexe" Funktion – später, wenn wir „händisch in unser Fenster malen", ist just dies die Stelle, wo.

```
// Wimp redraw window
void WimpRedrawWindow(void) {
    SWIRegs.r[1] = (int)(WimpPollBlock);
    SWI(Wimp_RedrawWindow,"Wimp_RedrawWindow");
    while(SWIRegs.r[0] != 0) {
        SWI(Wimp_GetRectangle,"Wimp_GetRectangle");
    }
```

Obiger Kasten zeigt die Grundroutine für einen Redraw, die offenbar gar nichts ausgibt, sondern nur das Minimum von dem tut, was sie formal tun soll, sprich wenn das alles ist, setze man das Bit, dass das Wimp das Fenster alleine ausgeben kann und spare sich diesen Code.

Wenn, wird die Redrawroutine natürlich auch etwas ausgeben und daher codiere ich sie lieber wie folgt mit der Variable „Looping", da ich bei den noch einzufügenden Codefragmenten wahrscheinlich die SWIRegs brauchen werde – und wenn Du gar in so einem Code den WimpPollBlock für andere Dinge zweckentfremdest, müsstest Du den Inhalt für den nächsten Aufruf von Wimp_GetRectangle natürlich auch wiederherstellen, aber sowas tut man nicht, da der Block, wie sein Namen schon suggeriert, nur für den Wimp-Poll gedacht ist.

Die Redraw-Funktion findest Du auf der nächten Seite oben.

```
void WimpRedrawWindow(void) {
    int Looping;
    SWIRegs.r[1] = (int)(WimpPollBlock);
    SWI(Wimp_RedrawWindow,"Wimp_RedrawWindow");
    // Note that Wimp_RedrawWindow implicitly
    // calls the first Wimp_GetRectangle for me
    Looping = SWIRegs.r[0];
    while(Looping != 0) {
        // Hier kommt der Code rein, der was tut...
        SWIRegs.r[1] = (int)(WimpPollBlock);
        SWI(Wimp_GetRectangle,"Wimp_GetRectangle");
        Looping = SWIRegs.r[0];
    }
```

Im zentralen Wimp_Poll-Loop wird der Teil, der sich um Fenster kümmert um einen Teil ergänzt:

```
// Window stuff
case ReasonCode_RedrawWindow;
    WimpRedrawWindow();
    break;
case ReasonCode_OpenWindow:
    WimpOpenWindow();
    break;
case ReasonCode_CloseWindow:
    WimpCloseWindow();
    break;
```

Habe fertig

So, nun wissen wir, wie man ein Fenster erzeugt, dass seinen Namen gefühlt verdient, da es einen Blick auf einen Ausschnitt der Dinge bietet, die das Programm präsentiert und der geneigte User kann das Fenster durch die Gegend schieben, scrollen, vergrößern und verkleinern.

Im nächsten Schritt spielen wir im Redraw-Teil ein wenig herum…

Ursprung

Die linke untere Ecke hat bei unserer Workarea die Koordinaten (0,0) und die X-Achse ist in der Horizontalen, wohingegen Y in der Vertikalen lebt – also ein Layout, dass viele sicherlich noch von Analysis aus der Schule kennen…

Theoretisch kann man, wie oben schon angedeutet, im Wimp problemlos (0,0) nach links oben verlagern, indem man die Workarea nicht so

```
WindowHeader->Workarea.MinX = 0;
WindowHeader->Workarea.MinY = 0;
WindowHeader->Workarea.MaxX = MaxX;
WindowHeader->Workarea.MaxY = MaxY;
```

sondern so

```
WindowHeader->Workarea.MinX = 0;
WindowHeader->Workarea.MinY = -MaxY;
WindowHeader->Workarea.MaxX = MaxX;
WindowHeader->Workarea.MaxY = 0;
```

initialisiert (bitte das Minuszeichen bei MinY beachten). Das funktioniert an sich problemlos und auch das mit dem Scrolloffset tut, wie erwartet und mir scheint, für einige ist links oben als „Ursprung“ des Fensters irgendwie passender, weil das z. B. bei einem Texteditor dem Anfang des Textes entspricht.

Doch wenn Du so operieren willst, musst Du das konsequent durchziehen – also auch bei Icons im Hinterkopf behaltend.

DownAndUp

Ich habe das Programm UpAndDown mal von links unten auf links oben für (0,0) der Workarea umgestellt – das Ergebnis ist DownAndUp. War an sich nicht so wild, da es genügte, folgende Änderungen zu machen:

Beim Anlegen eines Fensters ist die Workarea nun in der vertikalen nicht von 0 bis MaxY, sondern von -MaxY bis 0:

```
t_Window *CreateBasicWindow(char *Title,unsigned Flags,int Icons,int MaxX,int MaxY) {
    WindowHeader->Workarea.MinX = 0;
    WindowHeader->Workarea.MinY = -MaxY; // Links oben = (0,0)
    WindowHeader->Workarea.MaxX = MaxX;
    WindowHeader->Workarea.MaxY = 0;     // Links oben = (0,0)
}
```

Damit „kaschiere“ ist das mit den negativen Y-Werten hier und auch in den folgenden Fuktionen, da das einfach bequemer im Hauptprogramm ist.

Beim Fensteranlegen etc. ist der initiale Scrolloffset entsprechend Null:

```
t_Window *CreateBigWindow(char *Title,
    unsigned Flags, int Icons,
    int MaxX, int MaxY,
    int VisX, int VisY) {
    ...
    WindowHeader->ScrollY = 0;
    ...
}
```

Und die Icons positioniere ich ebenfalls mit negativen Koordinaten und negativer Höhe:

```
void FillTextIcon(t_Icon *Icon, char* Text,
    int White, int X, int Y,
    int Width, int Height) {
    ...
    Icon->BoundingBox.MinY = -Y;
    Icon->BoundingBox.MaxY = -Y-Height;
    ...
}
```

Das genügt schon, damit (0,0) links oben ist – aber die Ästhetik fordert noch eine kleine Korrektur, da ich so quasi oben und unten vertauscht habe und somit die vier Ecksymbole entsprechend anders beschriften darf, indem ich aus dem „U“ ein „O“ mache und umgekehrt – die anderen Parameter bleiben hingegen unverändert:

```
FillTextIcon(
    &(MyMainWindow->WindowIcon[0]),
    "LU",True,20,1000-44-20,260,44);
FillTextIcon(
    &(MyMainWindow->WindowIcon[1]),
    "LO",True,20,20,260,44);
FillTextIcon(
    &(MyMainWindow->WindowIcon[2]),
    "RU",True,1200-260-20,1000-44-
    20,260,44);
FillTextIcon(
    &(MyMainWindow->WindowIcon[3]),
    "RO",True,1200-260-20,20,260,44);
```

Über Kopf

Kann man so machen – muss man nicht – sollte aber!

Auch, wenn es für die eigene Vorstellungskraft einfacher ist, wenn links unten (0,0) ist und somit die Koordinaten in X- und Y-Richtung im Fenster positiv sind, spricht eine Sache massiv dafür, doch (0,0) nach links oben zu legen und damit innerhalb des Fensters für Y mit negativen Koordinaten zu arbeiten: Der Scrolloffset.

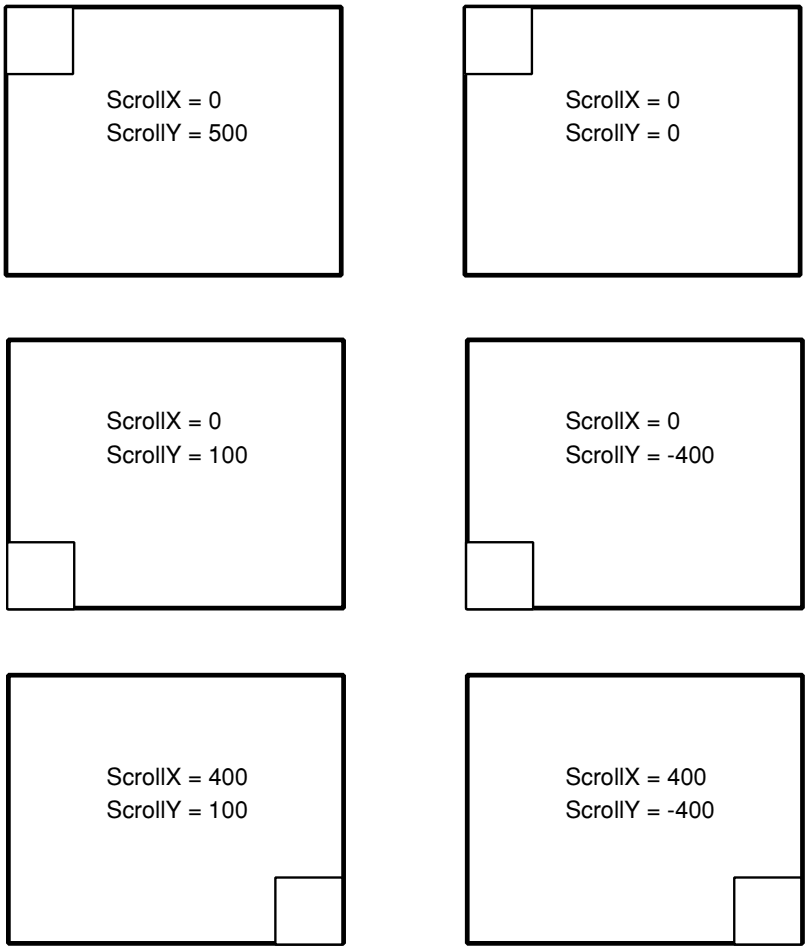
Dieser Offset bezieht sich gemeinerweise nicht auf den Ursprung des Fensters, sondern auf die linke und obere Seite des Fensters – keine Ahnung, wer da gerade einen lustigen Moment hatte, aber so ist das.

Konkret bedeutet das, dass sich ScrollX auf *Min*X aber ScrollY auf *Max*Y bezieht – da muss man schon aufpassen, aber es wird noch irritierender.

Nehmen wir eine Workarea mit einer Ausdehnung von 500 OS-Units in der Breite und Höhe – soweit noch einfach. Das Fenster dazu lassen wir mal 100x100 sein und dann schauen wir mal was wir als Scrolloffsets bekommen, wenn unten links (0,0) ist.

Zeigt das Fenster die linke, obere Ecke ist der Scrolloffset in X-Richtung 0, in Y-Richtung natürlich 500, weil die Fensteroberkante bei 500 liegt. Schiebe ich das Fenster nach ganz unten, wird der Scroll-Offset für Y zu 100 (ist ja auch irgendwie logisch, da die Fensteroberkante des 100x100er-Fenster nun 100 oberhalb der Nulllinie liegt). Schiebe ich es nach rechts wird die X-Richtung zu 400. An sich total logisch, da die Scrolloffsets im Kern sagen, wo die linke obere Fensterecke relativ zum Ursprung stehen. Anders formuliert ist der Scrolloffset in der horizontalen von 0-400, in der vertikalen aber von 100-500 – irgendwie fühlt sich das komisch an… finde ich.

Wenn ich aber (0,0) nach links oben packe, sprich mein Fenster geht von (0,0) bis (500,-500) und zeigt mein Fenster den Teil links oben sind beide Offsets 0; ist das Fenster rechts unten sind die Offsets 400 für X und -400 für Y. Das ist zumindest für mein Gehirn irgendwie einfacher, da die absoluten Scrolloffsets in beide Richtungen dieselben sind und ist das Fenster in der Ecke, wo (0,0) der Workarea ist, sind beide Offsets Null.



Scrolloffsets, wenn links unten (0,0) ist.

Scrolloffsets, wenn links oben (0,0) ist.

(0,0) = links oben

Also werde ich ab sofort meine Fenster so anlegen, dass (0,0) die linke obere Ecke der Arbeitsfläche ist und somit der Scrolloffset, der sich auf die linke obere Ecke des Fensters bezieht, dazu passt. Und bei Icons mache ich es natürlich analog, sprich die bekommen ihre linke obere Ecke ebenfalls als Ursprung für deren Positionierung im Fenster – ist ja klar.

Genaugenommen hätte ich von Anfang an den Fensterursprung nach oben links packen können und somit mit negativer Ausdehnung in der Vertikalen arbeiten können – und das hatte ich auch überlegt, dann aber beschlossen, erst einmal mit der Version, die sich „natürlicher“ anfühlt (sprich mit positiven Koordinaten im Fenster in X- und Y-Richtung) zu starten und den Schwenk zu machen, wenn es einen erkennbaren, guten Grund gibt.

Dabei „verstecke“ ich den Umstand, dass ich es in der Vertikalen mit nunmehr negativen Werten zu tun habe, in den verschiedenen Funktionen, sprich im eigentlichen Hauptprogramm ist links oben (0,0) und rechts unten (MaxX,MaxY).

Download

Selbstverständlich findest Du die beiden Programme UpAndDown.c und DownAndUp.c nebst Buggy.h und passendem Wimpy.h samt der Make- und Taskobeydateien als Download auf www.gag.de/gag-news/codeschnipsel; dort werden nach und nach auch die anderen Beispiele auftauchen. ❍

Fenster mit ein paar Zeilen

C-Programmierung unter RISC OS #14

Herbert zur Nedden

Die Macher von RISC OS haben in meinen Augen eine Unterlassungssünde begangen: Es gibt für Fenster kein sinnvolles mehrzeiliges Textobjekt, sprich will ich zehn Zeilen Text anzeigen, ist Handarbeit vonnöten.

Theoretisch kann ein Texticon mehrere Textzeilen beinhalten und per Validation-String-Kommando „L“ kann man die Darstellung ein wenig steuern, aber das ist nur für lange, einzeilige Texte gedacht, die nur umgebrochen angezeigt werden, sprich solche Kleinigkeiten wie Zeilenschaltungen werden hier nicht beachtet, so dass man die wenn, dann bestenfalls nur durch Einfügen der passenden Anzahl von Leerzeichen irgendwie simulieren kann, was nicht wirklich praktikabel ist (vor allem nicht, wenn ich einen Font nutze, bei dem nicht alle Zeichen dieselbe Breite haben).

Wenn ich es korrekt erkenne, kann die sog. Toolbox (das ist eine Sammlung von Modulen für die Wimp-Programmierung, die seit einiger Zeit zu RISC OS dazugehört) helfen, aber das wäre an dieser Stelle eine unnötige Komplexität, da ich dann die Toolbox einbinden müsste. Alternativ kann man aber auch einfach auf den Fensterhintergrund Text ausgeben… und das klappt, wenn auch nicht ganz ohne Überraschungen.

Zeilenstapel

Ich möchte ein Fenster haben, dass mir Platz für eine bestimmte Anzahl von Textzeilen in gegebener Länge bietet.

Damit der Code lesbarer ist, beginne ich hiermit, da ich ein Fenster mit 12 Textzeilen à 600 OS-Units Länge haben möchte:

```
#define TextIconCount  12
#define TextIconLength 600
```

Da ich die Icons im 20er-Abstand vom Fensterrand haben möchte, brauche ich somit ein Fenster, dass 640 breit ist – sprich 40 + TextIconLength.

Bei der Höhe kommt dazu, dass ich die Icons die üblichen 44 hoch sind und nicht aneinanderkleben, sondern einen kleinen 4er-Abstand bekommen sollen. Damit ist klar, dass das Fenster eine Höhe von 40 + TextIconCount*44 + (TextIconCount-1)*4 bekommen soll.

Anmerkung: In einem realen Programm, das mittels Icons Texte visualisiert, dürfte man sicherlich auf den Abstand zwischen den Icons verzichten, damit es nach einer einfachen Textausgabe ausschaut. Aber in diesem Codebeispiel sollen die separaten Icons im Fenster einzeln erkennbar sein.

Initial soll sich mein Fenster diskret als 200x100er präsentieren und somit habe ich diesen Aufruf:

```
MyMainWindow = CreateBigWindow((char*)MyTaskName,WindowFlags_AllToolsWindowRedraw,
    TextIconCount,40+TextIconLength,36+(TextIconCount*48),200,100);
```

Jetzt gilt es das dreckige Duzend zu befüllen und damit ich sehe, dass das auch klappt, soll in das oberste ein „A“, in das zweite ein „B“ usw. geschrieben werden. Da die Icons indirekt sind, sprich einen Zeiger auf den Text enthalten (immerhin soll da später mal mehr rein), ist klar, dass folgender Code in allen denselben Buchstaben präsentiert, da alle Icons auf denselben Text zeigen:

```
int i;
char Buf[2];
Buf[0] = 'A'; Buf[1] = 0; // Buchstabe und Stringende-Null
for(i=0;i<TextIconCount;i++) {
    FillTextIcon(&(MyMainWindow->WindowIcon[i]),(char*)(&Buf),
                 True,20,20+i*48,TextIconLength,44);
    Buf[0] = Buf[0] +1;
}
```

Also mache ich es so:

```
int i;
char Buf[TextIconCount*2];
for(i=0;i<TextIconCount;i++) {
    Buf[2*i] = (char)('A'+i); Buf[2*i+1]=0; // Buchstabe und Stringende-Null
    FillTextIcon(&(MyMainWindow->WindowIcon[i]),(char*)(&Buf[2*i]),
                 True,20,20+i*48,TextIconLength,44);
}
```

Mit dem Code im Programm bekomme ich ein Fenster in dem ein Duzend Icons mit weißem Hintergrund über die volle Breite (abzüglich des kleinen Randes) gehen und in denen „A“, „B“, …, „L“ stehen.

Netter Nebeneffekt dessen, dass (0,0) nun links oben ist und die Koordinaten in der Horizon- und Vertikalen aus Programmsicht positiv sind, so dass zur Berechnung der Position in der Vertikalen ein 20+i*48 genügt – wäre (0,0) links unten, wäre die Formel natürlich 20+(TextIconCount-i-1)*48.

Um dieser Einöde zu ein wenig Abwechslung zu verhelfen, tausche ich mal einen Wert aus:

```
char Bla[] = „This is a nice string“;
MyMainWindow->WindowIcon[4].
    IconData.IconTextIndirect.Text
    = (char*)(&Bla);
```

Dies zeigt auch, dass es hilfreich ist, dass die Icons indirekt sind, da mir das längere Texte gestattet.

Zeilenfläche

Nun probieren wir mal den alternativen Ansatz: Wir schreiben einfach auf den Fensterhintergrund ohne Icon – und das kann natürlich nur klappen, wenn wir das auch beim Redraw machen.

Mein Fenster hat somit gar keine Icons außer denen, auf dem Fensterrand.

```
MyMainWindow = CreateBigWindow(
    (char*)MyTaskName,
    WindowFlags_AllToolsWindowRedraw,
    0,1200,1000,800,400);
```

Erst einmal definiere ich mir eine globale Zählervariable MyCounter – die dient nur dazu, dass ich bei jedem Redraw-Aufruf in der ersten Textzeile sehe, der wievielte Durchlauf es ist.

Dann brauche ich noch den Code, der beim Redraw den Text ausgibt – aber bitte nur im entsprechenden Fenster:

```
void RedrawContent(t_WindowHandle Window) {
    if (Window == MyMainWindowHandle) {
        // Clear screen and put cursor to home position
        SWIRegs.r[0] = 12;  SWI(OS_WriteC,"OS_WriteC");
        // Print some text
        printf("*******%d*******\n",MyCounter++);
        printf("%s\n","Fangen wir mal langsam an");
        printf("%s\n"," und machen weiter");
        etc. // es folgen noch ein paar printf-Aufrufe
    }
}
```

Damit das zum Zuge kommt rufe ich den obigen Code in meiner Redraw-Routine:

```
// Wimp redraw window
void WimpRedrawWindow(void) {
    int Looping;
    SWIRegs.r[1] = (int)(WimpPollBlock);
    // Note that Wimp_RedrawWindow implicitly calls the first
    Wimp_GetRectangle for me
    SWI(Wimp_RedrawWindow,'Wimp_RedrawWindow");
    Looping = SWIRegs.r[0];
    while(Looping != 0) {
        RedrawContent(WimpPollBlock->RedrawWindow.WindowHandle);
        SWIRegs.r[1] = (int)(WimpPollBlock);
        SWI(Wimp_GetRectangle,'Wimp_GetRectangle");
        Looping = SWIRegs.r[0];
    }
}
```

Das Senden des Zeichencodes 12 via OS_WriteC positioniert den Cursor erst einmal nach links oben – alternativ kann ich auch den Code 30 senden. Die 12 löscht zusätzlich den Bildschirm (sprich das Fenster, was an sich nicht notwendig ist, weil das Wimp das eh schon tut), die 30 nicht.

Dann gebe ich konventionell ein paar Zeilen aus. Und siehe da, es klappt auch. Herrlich einfach – ich war begeistert – einen kurzen Moment.

Verschieben des Fensters klappt einwandfrei, aber wenn ich die Größe ändere, Scrolle usw. dann zeigt sich schnell, dass das Positionieren des Cursors nach oben links nicht meine Workarea, sondern das aktuell per Redraw angeforderte Rechteck meint. Das hat anfangs natürlich gepasst, weil das Fenster just die linke obere Ecke der Workarea zeigte.

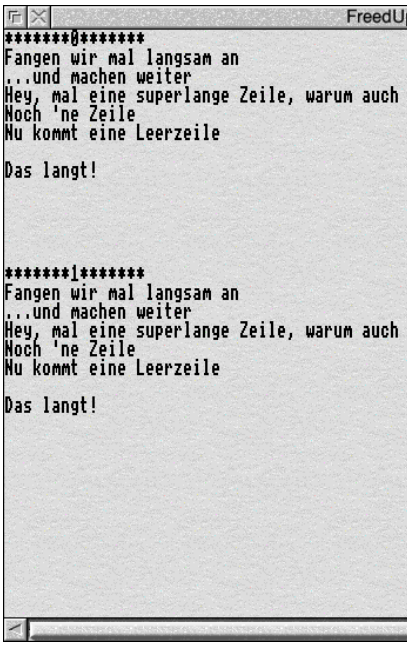
Mache ich das Fenster schnell höher sieht man, dass der in der ersten Runde ausgegebene Teil (der mit ***0***) unverändert erhalten geblieben ist, da das Wimp den Bildschirminhalt zur Hand hatte. Mein Programm musste nur den unteren Teil des Fensters neu zeichnen, da das erstmals in Erscheinung getreten ist – also bittet das Wimp um das entsprechende Rechteck und mein Code füllt das auch – und da ist nun gut zu erkennen, dass „Cursor Home“ den Textcursor in dem Rechteck und nicht in der Workarea positioniert, da die neue Ausgabe (mit ***1*** in der ersten Zeile) ein Stück weiter unten im Fenster landet, sprich in dem Rechteck, dass, da durch das Vergrößern des Fensters erschienen, nun erstmals zu zeichnen ist.

Was nebenbei auch erkennbar wurde ist, dass bei dieser Technik lange Zeilen nicht umgebrochen werden, aber damit hatte ich auch nicht wirklich gerechnet. Auch löscht der CLS nicht mein Fenster, sondern wenn, dann bestenfalls das Rechteck, denn sonst würde in obigem Screenshot der Inhalt der ersten Runde entschwunden sein.

Damit ist klar, dass dieser Ansatz einen eher begrenzten Reiz hat.

Globaler

Ich muss also das Problem mit der Positionierung des Cursors lösen, sprich erst einmal die Bildschirmposition der linken oberen Ecke meiner Workarea ermitteln, da der Text dazu relativ auszugeben ist.



Der SWI Wimp_GetRectangle liefert nicht nur die Maße des Fensters (sprich Visible), sondern auch den Scrolloffset. Die linke obere Ecke des Fensters mit Bildschirmkoordinaten (Visible.MinX, Visible.MaxY) entspricht den Koordinaten (ScrollX,ScrollY) der Workarea.

Damit ist es einfach, die Bildschirmkoordinaten der linken oberen Ecke der Workarea, sprich von dessen (0,0) zu errechnen:

```
X: Visible.MinX - ScrollX
Y: Visible.MaxX - ScrollY
```

Am Rande: Ich brauche ja die linke obere Ecke meiner Workarea und die habe ich so automatisch; wäre die linke untere Ecke die (0,0) müsste ich nun noch Workarea.MaxY auf die Y-Koordinate addieren, um zur linken oberen Ecke zu kommen – und da der Wimp_GetRectangle die Workarea-Daten nicht liefern, diese entweder erfragen oder mir im Programm merken. Wäre alles kein Problem, aber so ist es einfach schlanker (und einfacher zu verstehen).

Damit ist der Redraw-Code schnell klar: Positioniere an die o. g. Koordinaten und gib den Text aus im Vertrauen darauf, dass das Wimp dann nur den Teil der Ausgaben durchlässt, die im Fenster sichtbar sind, was es auch tut.

```
// Wimp redraw window
void WimpRedrawWindow(void) {
   int Looping;
   int OriginX, OriginY;
   SWIRegs.r[1] = (int)(WimpPollBlock);
   SWI(Wimp_RedrawWindow,"Wimp_RedrawWindow");
   // Note that Wimp_RedrawWindow implicitly calls the first Wimp_GetRectangle for me
   Looping = SWIRegs.r[0];
   while(Looping != 0) {
      // Get On-Screen position of top left corner of my work area (MinX,MaxY)
      OriginX = WimpPollBlock->RedrawWindow.Visible.MinX - WimpPollBlock->RedrawWindow.ScrollX;
      OriginY = WimpPollBlock->RedrawWindow.Visible.MaxY - WimpPollBlock->RedrawWindow.ScrollY;
      RedrawContent(WimpPollBlock->RedrawWindow.WindowHandle,OriginX,OriginY);
      SWIRegs.r[1] = (int)(WimpPollBlock);
      SWI(Wimp_GetRectangle,"Wimp_GetRectangle");
      Looping = SWIRegs.r[0];
   }
}
```

Der Redraw-Code oben im Kasten ist erst einmal übersichtlich. Was ich tue ist, die Bildschirmkoordinaten der linken, oberen Ecke des Fensters zu ermitteln und dann die Information an die RedrawContent-Routine zu senden. Den Handle des Fensters bekommt die Routine nur mit, damit sie nicht im falschen Fenster Dinge absondert.

Codetechnisch hätte ich auch einfach den WimpPollBlock als Ganzes an die RedrawContent-Routine geben können und dann die Origin-Berechnung dort machen.

Texten

Die eigentliche Ausgabe kann auf mindestens zweierlei Arten erfolgen. In Basic kann ich den Cursor mit MOVE x%,y% positionieren und dann den Text mit PRINT ausgeben, aber wir programmieren in C. Für die Ausgabe ist printf bestens gewappnet und den MOVE x%,y% leistet der SWI OS_Plot mit den Parametern 4, x% und y% – aber das tut nur, wenn man den OS_Plot vor jeder Zeile ausführt, da die Zeilenschaltung den Cursor zwar irgendwie in die neue Zeile packt, aber dann auch noch in das aktuelle Redraw-Rechteck und somit eher nicht unter die vorherige Zeile – siehe Kasten unten links auf dieser Seite.

Die Berechnung der Y-Koordinate so hardcodiert wie oben macht man in einem „richtigen“ Programm nicht so, aber da hat man auch nicht nur ein paar Demostrings, so dass man entweder eine Hilfsvariable dafür nutzt, die für jede Zeile um 44 verringert wird, oder eine Schleife à la for (i=0;i<Max;i++) und nimmt für die Vertikale Y-i*44.

Die Alternative, die obendrein, wenn man möchte, die Verwendung anderer Schrifttypen leicht möglich macht, ist die Nuzuung des SWI Font_Paint, mit dem es ebenfalls gut funktioniert.

```
void RedrawContent(t_WindowHandle Window, int X, int Y) {
   char Bl[] = "Fangen wir mal langsam an";
   char B2[] = "... und machen weiter";
   etc.
   if (Window == MyMainWindowHandle) {
      SWIRegs.r[0] = 4;
      SWIRegs.r[1] = X;
      SWIRegs.r[2] = Y-0*44;
      SWI(OS_Plot,"OS_Plot");
      printf("%s\n",Bl);
      SWIRegs.r[0] = 196;
      SWIRegs.r[1] = X;
      SWIRegs.r[2] = Y-1*44;
      SWI(OS_Plot,"OS_Plot");
      printf("%s\n",B2);
      etc.
   }
}
```

Font_Paint erwartet seinen Auftrag in acht Registern, wobei ich mir das Leben insofern einfach mache, dass ich einfach den Default-Font nutze und somit bei den Registern kaum etwas angeben muss, sprich das Gros auf Null setze.

In Register 2 kommt ein Flag und hier setze ich nur Bit 4, da ich für die Koordinaten OS-Units verwende. Register 3 und 4 enthalten die Position auf dem Schirm und Register 1 den Zeiger auf den Text.

Die Bildschirmposition initialisiere ich mit den übergebenen Koordinaten für den Ursprung meines Fensters und vor jeder Zeile, die ich ausgebe, ziehe ich 44 in der Vertikalen ab, sprich eine Zeilenhöhe, damit die Zeilen untereinander erscheinen – siehe Kasten unten rechts auf dieser Seite.

Hier nutze ich aus, dass der SWI Font_Paint die Registerwerte nicht verändert; sonst hätte ich jeweils alle Register bei jedem Aufruf setzen dürfen und, ähnlich wie bei OS_Plot mit printf natürlich für die Berechnung der Vertikalen entsprechend codiert.

Im Programm selbst gebe ich sieben Textzeilen und darunter auch längere aus, aber das Prinzip macht obiges sicher schon deutlich.

```
void RedrawContent(t_WindowHandle Window, int X, int Y) {
   char Bl[] = "Fangen wir mal langsam an";
   char B2[] = "...und machen weiter";
   etc.
   if (Window == MyMainWindowHandle) {
      // Clear screen and put cursor to home position
      SWIRegs.r[0] =  0;
      SWIRegs.r[2] = 0x10;  // OS Units
      SWIRegs.r[3] = X;
      SWIRegs.r[4] = Y;
      SWIRegs.r[5] = 0;
      SWIRegs.r[6] = 0;
      SWIRegs.r[7] = 0;

      SWIRegs.r[4] -= 44;
      SWIRegs.r[1] = (int)(&Bl);
      SWI(Font_Paint,"Font_Paint");
      SWIRegs.r[4] -= 44;
      SWIRegs.r[1] = (int)(&B2);
      SWI(Font_Paint,"Font_Paint");
      etc.
   }
}
```

Nicht oben links

Die Redraw-Routine wird mit den Bildschirmkoordinaten der linken oberen Fensterecke gerufen. Soll die Textausgabe nicht links oben erscheinen, sondern mit meinem gerne genommenen Rand von 20 OS-Units und oben im Fenster obendrein Platz für ein Texticon (Höhe 44 OS-Unit) lassend und von dem ebenfalls die bekannten 20 Abstand drüber haben, dann bietet sich etwas wie dies an:

```
PosX = X+20;
PosY = Y-20-44-20;
for (i=0;i<LineCount;i++) {
   PosY -= 44;
   OS_Plot oder Font_Paint mit PosX
   und PosY als Koordinaten
}
```

(Ja, man gewöhnt sich recht schnell daran, bei X-Koordinaten zu addieren, bei Y hingegen zu subtrahieren.)

PosX ist der linke Rand plus 20 für den Abstand von selbigem. Für PosY ziehen wir 20+44+20 von Y ab, um so einen Abstand, eine Textzeile und noch einen Abstand nach unten zu gehen.

Bei der Ausgabe der Textzeilen müssen wir nur immer vor dem Ausgaben 44 von PosY abziehen, um zur Grundlinie des Textes zu kommen.

Der Beispielcode weiter oben war absichtlich etwas mehr „zu Fuß“, damit das, was passieren soll, etwas transparenter ist.

Performance

Was noch anzumerken wäre ist, dass die Lösung mit einer Hand voll Texticons etwas performanter ist, als die, bei der ich den Text mit Font_Paint ausgebe. Das ist auch nicht verwunderlich, da bei den Icons das Wimp das alles problemlos alleine wuppt und somit auch recht effizient abarbeitet, wohingegen das Painten der Fonts bedeutet, dass mein Programm die volle Workarea beschreibt (also im Zweifelsfall viel zu viel absondert) und das Wimp dann die Ausgabe per clipping beschneiden darf. Die Version mit OS_Plot und printf hingegen ist ähnlich Performant, wie die Lösung mit Texticons, was wahrscheinlich daran liegt, dass es einfach rudimentärer ist, einen simplen printf abzusondern, als den Font_Paint.

Keine Sorge, der Performanceunterschied ist auf realer Hardware nicht wirklich merkbar, bei RPCEmu, dass dank Emulation der CPU schon träger ist (und bekanntlich schreibe ich die C-Programme für diese Serie mit dem RPCEmu, damit ich sicher bin, dass nicht irgendwelche Dinge, die auf meinem mini.m im System vorhanden sind benötigt werden, ohne, dass ich darauf hinweise), ist es merkbar.

Spezial oder nicht

Das Programm FreedUp kann beide Versionen der Textanzeige und per Mausklick auf das Symbolleistensymbol bei geschlossenem Fenster entscheidest Du, wie. Beim Rechtsklick wird Font_Paint genommen, beim Linksklick darf OS_Plot mit printf ran:

```
case ReasonCode_MouseClick:
   if (WimpPollBlock->MouseClick.Icon == MyBarIcon) {
      switch(WimpPollBlock->MouseClick.Buttons) {
      case MouseButtonRight:
        RedrawKind = "F";
        MyMainWindowHandle = WimpOpenNewWindow(MyMainWindow,MyMainWindowHandle,True);
        break;
      case MouseButtonMiddle: // Menu click
        OpenMenu(MyBarMenu);
        break;
      case MouseButtonLeft:
        RedrawKind = "P";
        MyMainWindowHandle = WimpOpenNewWindow(MyMainWindow,MyMainWindowHandle,True);
        break;
      }
   }
   break;
```

Kaum, dass ich in dem String ein CR oder LF eingefügt habe, sprich das Zeichen mit dem Code 13 oder 10, um so eine Zeilenschaltung zu erhalten, erschien auf dem Schirm nur der Teil des Strings davor – offenbar wird bei Text im Icon das erste Steuerzeichen als Stringende interpretiert. Vielleicht macht RISC OS das so, weil BASIC den Code 13 als Stringende verwendet, damit man bei BASIC-Programmen das Leben leichter hat. Also nehme man eine Schrift mit fester Laufweite (monospaced), damit es einfach ist und füge die passende Zahl von Leerzeichen ein, damit das mit dem Umbruch passt – nein, nicht wirklich.

Die Alternative, all diesen Iconüberbau zu umschiffen und den Text einfach ins Fenster zu schreiben funktioniert auch, wobei die Lösung mit OS_Plot und printf performancemäßig die Nase vorne hat, der Font_Paint hingegen mehr optische Flexibilität bietet.

Fazit

Das Anlegen und füllen von einem Texticon pro Zeile funktioniert ohne Probleme, wie im Programm LinedUp zu sehen: obald ich die Texticons ohne Rand drumherum und ohne Abstand zwischen ihnen positioniere, sieht der geneigte Anwender das nicht mehr.

Mit einem mehrzeiligen Texticon habe ich es auch probiert. Ich bekam so ein großes Icon, bei dem der Text linksbündig aber in der Vertikalen zentriert zu sehen war – komisch, ich habe in den Icon-Flags nicht um Zentrierung gebeten. Immerhin kann ich das Wimp dazu bringen, einen langen Text in so einem Icon umzubrechen, indem ich in den Validation-String (den wir später noch kennen lernen werden) ein „L40“ schreibe – das ist schon mal gut.

Es ist echt bedauerlich, dass in RISC OS kein Icon oder sonstiges Fensterobjekt Fenster vorgesehen ist, dass sinnvoll mehrere Textzeilen gestattet.

Download

Selbstverständlich findest Du wiederum LinedUp.c und FreedUp.c nebst Buggy.h und Wimpy.h samt der Make- und Taskobeydateien als Download auf www.gag.de/gag-news/codeschnipsel; dort werden nach und nach auch die anderen Beispiele auftauchen. ❍

(Ver)Laderoller

Kyocera und Punkte

Herbert zur Nedden

Die Ursache für die Pünktchen auf den Ausdruck in News 174 ist gefunden – erstaunlich, was ich dabei erlebte…

Aus dem Abstand der Pünktchen von 3,8cm war an sich klar, dass da etwas rundes im Spiel ist, dass einen Durchmesser von ca. 1,2cm hat – und Rollen sind in so einem Drucker reichlich vorhanden.

Der Support von Kyocera tippte erst auf die Trommel (deutlich dicker) und dann auf den Laderoller und schrieb mir noch – das ist der Klopper, finde ich, für ein internationales Unternehmen –, dass es sich bei meinem Drucker (den ich bei einem deutschen Händler erstanden habe) nicht um ein deutsches Produkt handele und sie ihre freiwillige Garantie nur auf ausgewählte Geräte, die sie als deutsche Organisation in den Handel gebracht haben, geben würden.

Der Händler, bei dem ich den Drucker vor knapp zwei Jahren gekauft habe, reagierte erst mal nicht, aber der Kyocera Servicepartner, an den ich mich wandte, da Kyocera meinte, ich solle mich an meinen Händler oder einen Servicepartner wenden, sehr schnell und vor allem kompetent, da sie den Entwickler oder Laderoller in Verdacht hätten.

Ich baute den Laderoller aus – der Durchmesser passt – und unter guter Beleuchtung konnte ich auf der schwarzen gummiartigen Rolle an passender Stelle zwei schwache Dellen entdecken. Neuer Laderoller – Drucker tut wieder gut. Ob der Entwickler ebenfalls eine 1,2cm-Rolle hat, habe ich daher nicht mehr nachgemessen, zumal es aufwändiger ist, den auszubauen, aber denkbar ist es schon.

Dem Händler schrieb ich erneut – dieses Mal in juristisch angehauchtem Deutsch mit Fristsetzung – und siehe da, sie antworteten prompt, allerdings mit dem Hinweis auf die Beweislastumkehr nach sechs Monaten, womit sie nicht Unrecht haben.

Gut versteckt habe ich dann auch bei Kyocera gefunden, dass der Laderoller ein Verschleißteil ist und somit eh von der üblichen Garantie ausgenommen – dass er nach nur rund 13.000 Seiten (also nicht einmal zwei Tonerkartuschen) hin ist, finde ich nicht berauschend, aber immerhin habe ich was gelernt und hoffe mal, dass der neue länger hält.

Der Drucker hat seinerzeit übrigens rund 900€ gekostet – der handelsübliche Preis liegt eher so bei dem Anderthalbfachen (der Händler hat mir den guten Preis seinerzeit damit erläutert, dass sie mal eine größere Charge erstanden hätte, um einen guten Einkaufspreis zu haben, und nun ihre Restbestände anbieten würden – dass Kyocera als internationales Unternehmen bei der Garantie so lokal orientiert ist, darauf wäre ich nie gekommen), so dass ich selbst nach Kauf des Laderollers preislich gut „davongekommen“ bin, denn brauchbare A3-Drucker mit Duplex sind teuer und die meisten obendrein riesig. Insgesamt bin ich mit dem eher handlichen Kyocera Ecosys P4040dn zufrieden.

Ende gut – alles gut. ❍

Altlasten?

!Cache - ein (zu) gutes Gedächtnis

Herbert zur Nedden

Für RISC OS gibt es eine Applikation namens !Cache, die u. a. NetSurf nutzen kann, um Seiten, die es mal geladen hat, lokal zu cachen. Früher zu Zeiten von Modems und auch in den Anfängen von DSL, wo man sich über 1MBit und somit immerhin viel mehr als das, was ISDN konnte, freute, waren solche Caches wichtig – alleine schon beim Zurückblättern machten sie den Unterschied, ob die Seite (relativ) schnell oder erst nach längerem Warten erschien.

Heutzutage, wo 16MBit und Flatrates als unterer Standard gelten hingegen, ist der Benefit von Caches deutlich übersichtlicher, zumal auf vielen Webseiten die omnipräsente Werbung zu cachen eh wenig nützt, da die gerne bei jedem Seitenaufbau eine andere ist, aber es gibt sie immer noch. In !NetSurf kannst Du natürlich einstellen, ob es !Cache nutzt und wie lange es Dinge dort vorhalten will.

Ein netter Nutzen von !Cache ist übrigens, dass man mal sehen kann, welche Unmengen von Dingen man so abruft, wenn man die eine oder andere Webseite besucht – zumal es wohl so ist, dass das mit dem Entsorgen historischen Inhaltes nicht immer klappt.

Auch haben einige User bemerkt, dass vor allem bei einem länger laufenden !Cache, in dem sich auch diverse Unterverzeichnisse ansammeln, die Performance leiden kann – und langsame Laufwerke wie z. B. billige, träge SD-Karten können ebenfalls diese Wirkung entfalten. Auf einem ARMX6 mit schneller SSD soll er ein wenig helfen, so R-Comp.

Andrew Rawnsley berichtete jedenfalls, dass bei ihm !Cache auf stattliche 33GB (ja, Giga, nicht Mega) angewachsen war – das ist der Moment, wo es Sinn macht, ihn mal händisch zu entleeren, da die Automatik augenscheinlich nicht griff. Eine praktische Nebenwirkung ist, dass dann Backups der Bootapplikation schneller sind, da !Cache in !Boot.Resources beheimatet ist, aber nicht sicherungswürdig ist.

In !NetSurf kann man den Cache via Symbolleistenmenü steuern – und mein Tipp wäre, den Cache nicht zu nutzen, da der eher wenig bringt, wenn überhaupt – aber besagte Nachteile haben kann. Auf Systemen, deren Laufwerk eine SD-Karte ist, würde ich !Cache sogar definitiv löschen, weil diese Kleinode bekanntlich eine limitiere Zahl von Schreibzugriffen verkraften, also vermeiden wir sie hier und verlegen !Scrap auf die RAM-Disc.

Vor dem Erscheinen von !Cache haben die Browser zum Teil in !Scrap ein wenig Historie vorgehalten, was man tun kann, aber in !Scrap sollte eigentlich nix landen, was man behalten möchte – wobei !NetSurf diesen Mist mit der Datei RUfl_cache leider immer noch tut, weshalb man dieses Kleinod, wenn man !Scrap auf die RAMDisc packt, beachten darf, damit !NetSurf sie nicht beim Start erst einmal neu aufbaut, was Zeit kostet. ❍

SD-Schonung

!Scrap im RAM

Herbert zur Nedden

Das Verlagern von !Scrap auf die RAM-Disc macht Sinn, da es die Zugriffe darauf beschleunigt und vor allem bei Systemen mit der Bootapplikation auf einer SD-Karte deren Lebenserwartung erhöht. Auch haben die neueren ARM-Rechnerchen i. a. so viel RAM, das man eh meist das Gros davon sonst ungenutzt lässt.

Folgende kleine Obey-Datei, die nach PreDesk gehört und am besten direkt nach DiscSetup geladen wird (also z. B. DiscSetupScrap genannt wird), macht es sogar noch schlanker, als meine alte Lösung, da sie kein Muster-!Scrap kopiert, sondern einfach nur die notwendigen Verzeichnisse anlegt, Systemvariablen setzt und natürlich, so vorhanden, die RUfl_cache von !NetSurf dort hinpackt.

Die beiden Variablen TempS für das Scrap auf der RAM-Disc und TempC für die RUfl_cache von NetSurf habe ich nur definiert, damit die Kommandos übersichtlicher sind.

Ob es notwendig ist, ein Scrap-Verzeichnis mit da drin dann ScrapDir anzulegen, oder auch das untere Skript genügt, habe ich nicht probiert. Vermutlich funktioniert das auch und diese Zweistufigkeit kommt nur daher, dass !Scrap im Original eine Applikation mit !Boot, !Run und !Sprites ist, so das ein Unterverzeichnis ScrapDir für die Daten sicherstellt, dass da nix falsches überschrieben wird.

Auf virtuellen RISC OS-Systemen würde ich !Scrap übrigens nicht in die RAM-Disc schieben, denn da ist die Platte tendenziell eh schnell und keine SD-Karte oder so. Auch ist die mögliche RAM-Menge bei Emulatoren meist weniger üppig und das Wirtsbetriebssystem braucht auch etwas davon.

Einen kleinen Nachteil von !Scrap auf der RAM-Disk möchte ich nicht verschweigen: Beim Shutdown fragt RISC OS nach, weil die RAM-Disk nicht leer ist, ob man wirklich will, aber ein schnelles !Scrap ist mir lieber. ❍

```
IfThere RAM::RamDisc0.$ Then Else Obey
Set TempS RAM::RamDisc0. $.Scrap
Set TempC <Boot$Dir>.Resources.!Scrap.ScrapDirs.ScrapDir.RUfl_cache
CDir <TempS>
CDir <TempS>.ScrapDir
IfThere <TempC> then Copy <TempC> <TempS>.ScrapDir.RUfl_cache A~C~DF~L~N~PQ~R~S~T~V
Set Scrap$Dir <TempS>
Set Wimp$ScrapDir <TempS>.ScrapDir
Set Wimp$Scrap <Wimp$ScrapDir>.ScrapFile
UnSet TempC
UnSet TempS
```

Scrap auf die RAM-Disc packen

```
IfThere RAM::RamDisc0.$ Then Else Obey
Set TempS RAM::RamDisc0. $.Scrap
Set TempC <Boot$Dir>.Resources.!Scrap.ScrapDirs.ScrapDir.RUfl_cache
CDir <TempS>
IfThere <TempC> then Copy <TempC> <TempS>.RUfl_cache A~C~DF~L~N~PQ~R~S~T~V
Set Scrap$Dir <TempS>
Set Wimp$ScrapDir <TempS>
Set Wimp$Scrap <Wimp$ScrapDir>.ScrapFile
UnSet TempC
UnSet TempS
```

Scrap auf die RAM-Disc aber mit weniger Tiefgang

JPG und JPEG

Sprechende Dateitypen

Herbert zur Nedden

Unter RISC OS sind die Dateitypen bekanntlich nicht sichtbar im Dateinamen enthalten, sondern als Attribut „versteckt“, so dass wir „Foto1“ als Dateinamen nehmen können, statt „Foto1/JPG“, die Datei aber trotzdem als JPEG-Bild daherkommt. Speziell auf Altsystemen, wo noch Dateinamen auf üppige zehn Zeichen limitiert sind, ein echter Segen, da immerhin so rechnerisch vier Zeichen mehr für den Namen verfügbar sind.

Sofern sichtbar im Dateinamen, wie unter Windows üblich, ist das Gros dieser Namenssuffixe dreistellig – wohl, weil früher Dateinamen im 8+3er-Format mehr nicht boten. Aber dann kamen die langen Namen und damit die Flexibilität und ein wenig Wildwuchs, da es nunmehr für einige Formate gleich mehrere Dateinamensendungen gibt wie z. B. JPG und JPEG oder HTM und HTML, sprich die alte dreistellige lebt friedlich neben der vierstelligen, die quasi den Typ ungekürzt wiedergibt.

File$Type

Da der Dateityp unter RISC OS aber keine Zeichenfolge, sondern eine dreistellige Hexzahl (genauer ein zwölfbittiger Wert, den man als dreistellige Hexzahl darstellen kann) wie z. B. die &C85 für JPEG ist, gibt es nur einen Wert für einen Dateitypen. Damit der geneigte Anwender nicht mit diesen kryptischen Hexwerten operieren muss, kann man zu diesen einen sprechenden Namen via Systemvariable hinterlegen:

```
Set File$Type_C85 JPG
```

Damit versteht RISC OS „JPG“ als Alias für &C85 beim Dateityp und somit für JPEG-Bilder, aber halt nicht „JPEG“, aber das ist schnell erledigt:

```
Set File$Type_C85 JPEG
```

Schon kennt RISC OS den Dateityp „JPEG“ aber dafür nicht mehr „JPG“, sprich es gibt nur eine Textform per Default – und da die obendrein in einer Systemvariablen steht, ist sie leicht änderbar (wie man bekanntlich auch leicht das Sprite austauschen kann, mit dem der Filer Dateien eines Typs darstellt), weshalb Programme in ihrem Code auf den Hexwert prüfen (sollten), denn der ist wohldefiniert und erspart dem Programm, den Umweg über irgendwelche Systemvariablen zu gehen.

LanManFS

Wenn ich mit LanManFS oder so ähnlichem auf entfernte Datenträger zugreife, dann kommt MimeMap ins Spiel – das hat u. a. die Aufgabe, zwischen Dateinamensendungen und RISC OS-Dateitypen zu mappen und hier findet man i. a. das Mapping von .jpg und von .jpeg zu &C85, sprich MimeMap kennt beide Endungen – perfekt.

Somit zeigt mir LanManFS eine Datei als JPEG an, egal, ob sie unter Windows oder was-auch-immer als „Bild.jpg“ oder „Bild.jpeg“ abgelegt ist.

Und sichere ich ein JPEG-Bild dorthin – genauer: sichere ich eine Datei mit dem RISC OS-Dateityp &C85 und benenne die Datei mit passender Endung wie „Bild/jpg“ oder „Bild/jpeg“, landet sie mit just dem Namen „Bild.jpg“ bzw. „Bild.jpeg“ auf der Platte.

Ja, gut erkannt aber Dir auch sicher bekannt: Beim Mainstream ist der Punkt ein erlaubtes Zeichen in Dateinamen und dient als Trenner vor der Namensendung, wohingegen unter RISC OS der Punkt der Verzeichnistrenner ist, so dass wir in Dateinamen statt des Mainstream-Punktes den Schrägstrich verwenden dürfen.

Hat meine Datei keine oder eine andere Endung, hängt LanManFS beim Sichern den RISC OS-Dateityp dran, so dass ich „Bild,c85“ oder auch gar „Bild.jepg,c85“ erhalte, wenn ich die Datei als „Bild“ (sprich ohne Endung) oder „Bild/jepg“ (mit unpassender Endung) sichere, womit der RISC OS-Dateityp erhalten bleibt.

Soweit also eine saubere Sache, aber… was ist, wenn ich unter RISC OS beim Setzen des Dateityps JPG und JPEG beides verwenden können möchte, weil es praktischer ist oder ich mich einfach nicht entscheiden will?

File$Types

Thomas Milius hat sich den Sourcecode von RISC OS angesehen und hier ist die Lösung:

```
Set File$Type_C85 JPG
Set File$Type_0C85 JPEG
```

Bei der Anzeige des Dateityps im Filer „gewinnt“ bei hierbei der normale Eintrag, also das „JPG“, aber ich kann nun auch einer Datei den Typ „JPEG“ geben und das erwartete passiert.

Grund, dass das klappt ist, wie RISC OS den Teil hinter File$Type_ auswertet, um die Zeichenfolge in die dreistellige Hexzahl zu wandeln. Der Code nutzt intern eine etwas allgemeiner funktionierende Konvertierungsroutine und daher kann man den Hexwert entsprechend flexibler spezifizieren.

Aber auch dies klappt:

```
Set File$Type_&C85 JPEG
Set File$Type_16_C85 JPEG
```

Ersteres macht per „&“ klar, Hex; letzteres hingegen spezifiziert die Zahlenbasis 16 (Hex) explizit.

Hier noch ein paar Variationen des Themas:

```
Set File$Type_FFF Text
Set File$Type_0FFF Txt
Set File$Type_4_333333 Hihi
Set File$Type_2_lllllllllll Blabla
```

Nun kann ich die Dateitypen „Txt“, „Hihi“ und „Blabla“ verwenden, um einer Datei den Typ Text (&FFF) zu verpassen – angezeigt wird er dann im Filer mit dem Default, sprich „Text“.

Anzumerken wäre noch, dass es einige Stellen wie IP-Adressen gibt, wo eine führende Null impliziert, dass die Zahl oktal und nicht dezimal verstanden wird, sprich die IP-Adresse 1.1.1.11 ist nicht dasselbe wie 1.1.1.011, aber bei File$Type wird die Angabe, wenn die Zahlenbasis nicht explizit genannt wurde, als Hexzahl interpretiert, sprich ein File$Type_077 meint, wie auch File$Type_77 den Dateityp &77. Aber hier ist ja auch der interne Default Hex und nicht Dezimal/Oktal.

Deutsch

Warum dieses Feature nicht längst verwendet wurde, um schadfrei ein wenig mehr Deutsch in ein deutsches RISC OS zu bringen, aber trotzdem die Originaldateitypnamen vorsorglich zu erhalten, ist mir nicht klar. Hier mal eine Anregung:

```
set File$Type_DDC Archiv
set File$Type_0DDC Archive
set File$Type_FAE Ressource
set File$Type_0FAE Resource
set File$Type_FCC Gerät
set File$Type_0FCC Device
```

Und noch ein paar (nur die deutschen):

```
set File$Type_FD7 TaskBefehl
set File$Type_FEB Befehl
set File$Type_FF2 Konfiguration
set File$Type_FF4 Druckausgabe
set File$Type_0FF4 Printout
```

RPCEMu

Kleiner Exkurs zu RPCEmu, da das MimeMap nicht nutzt. Wie oben zu lesen, mappt MimeMap z. B. die beiden Endungen “.jpg“ und “.jpeg“ zu &C85 und damit ist an sich die Welt in Ordnung. Doch sobald sich die Konfiguration von MimeMap ändert, besteht das Risiko, dass dieses Mapping verändert wird oder auf der Strecke bleibt – für gängige Dateitypen an sich unwahrscheinlich, aber immerhin denkbar. Nein, mehr als denkbar, denn wenn Du ein älteres RISC OS unter RPCEmu verwendest, dass noch mit der Vorgängertechnik DosMap arbeitet, ist die Wahrscheinlichkeit, dass es Unterschiede gibt, plötzlich größer.

Daher haben die Macher von RPCEmu beschlossen, dass ihr HostFS dieses Mapping gar nicht macht, um den RISC OS-Dateityp sicher zu erhalten, egal, welches RISC OS mit welchen Settings da gerade verwendet wird, damit ein OS-Wechsel möglichst risikoarm möglich ist (und wenn es knallt, dann nicht wegen nicht-gemappter Dateitypen).

Das ist ein durchaus sinnvoller Ansatz – schön wäre gewesen, wenn man dann innerhalb von RPCEmu zumindest für ein paar Dateitypen HostFS das Mapping hätte explizit beipulen können, aber da LanManFS auch bei dem einfachen Netzwerk via NAT (also ohne die Installation von TAP-Interfaces auf dem Wirt) funktioniert, nutze einfach LanManFS für die Dateien, für die Du das Mapping haben möchtest, damit die „,xxx“-Endungen den Wirt nicht irritieren.

Fazit

Die Tipps, wie Du unter RISC OS gleichzeitig „JPG“ und „JPEG“ als Dateityp verwenden kannst, verdanken wir RISC OS Open, wo der Quellcode von RISC OS einsehbar ist, und Thomas Milius, der genau das tat und damit herausfand, dass Set File$Type_0C85 JPEG funktioniert. Danke. ❍

Gemeinsam sind wir stark

Shared Libraries

Herbert zur Nedden

Die Programmiersprache C hat diverse Funktionen eingebaut und damit man deren Laufzeitcode nicht in jedem Programm fest drin haben muss, ist es üblich, diese als sog. shared Library vorzuhalten. Das macht nicht nur die Programme kleiner, sondern spart auch Hauptspeicher, wenn bestimmte Codeteile nur einmal im RAM liegen, statt mehrere Male. Zugegeben, heutzutage, wo RAM tendenziell reichlicher verfügbar ist, ist das weniger relevant, aber warum damit unnötig aasen.

Ein weiterer Vorteil dieser Libraries ist, dass im Falle von Korrekturen (bei gut abgehangenen Libraries eher seltener) und vor allem beim Beheben von Sicherheitslücken (die auch in uraltem Code gefunden werden), die shared Library ausgewechselt werden kann und schon nutzen alle Anwendungen den neuen, verbesserten Code. Das ist dann doch schon relevant.

Unter RISC OS kommen zwei Aspekte dazu: Verbesserte Portierungen und der Support von Features neuerer ARMe.

CLib

Unter RISC OS wurde das für C so implementiert, dass die Shared C Library als Modul daherkommt und man beim Programmieren lediglich die Library Stubs anziehen musste, die als Adapter zwischen meinem C-Programm und dem Code in der Shared C Library dient (Stubs setzt also, platt gesagt, die Standard-C-Funktionen in die entsprechenden Modulaufrufe um).

Ein wenig „lustig“ war es dann eine Zeit lang, als der Wechsel von 26 zu 32 Bit kam, der natürlich an dieser Library nicht spurlos vorüberging. In der Zeit gab es somit nicht nur das Modul mit der Shared C Library in zwei Inkarnationen, sondern auch von Stubs entsprechende Varianten.

Dabei bitte ich im Hinterkopf zu behalten, dass beim 26bittigen Betrieb die Prozessorflags im selben Register wie der Program Counter stehen und somit beim Sichern desselben auf dem Stack beim Aufruf einer Unterroutine automatisch mit gesichert und beim Rücksprung wiederhergestellt wurden. Im 32-Bit-Betrieb hingegen sind die Flags in einem separaten Register. Das ist programmcodetechnisch leicht adressierbar, musste aber halt gemacht werden, wenn z. B. Flags sicher erhalten (also extra gesichert und restauriert) bzw. mit bestimmten Werten zurückgegeben werden sollten, da die nun anderswo lagen… oder nicht.

Doch dauerte es nicht lange, und es gab eine Variante von Stubs, die mit der 26- und der 32-bittigen Shared C Library harmonierte und die Welt war für Programmierer wieder einfach. Und für die User auch, da sie die zu ihrem System passende CLib nutzen konnten.

Und da die Shared C Library „nur“ diverse Standard-C-Funktionen bereitstellt, ist ihr Funktionsumfang sauber und klar definiert – wie praktisch.

Linux

In der Linux-Welt gibt es ebenfalls shared Libraries – und zwar einige. Aber da ist die Implementierung ein wenig anders vom Konzept her, da es vorgesehen ist, dass man von *einer* solchen Library mehrere verschiedene Versionen parallel haben kann – die Applikation schnappt sich entweder die aktuelle oder eine in einer bestimmten Version. Die aktuelle ist dazu via versionsnummerlosem symbolischem Link zu finden.

Ein guter Grund für Applikationen nicht die brandneue Library zu nutzen, sondern eine bestimmte Version, kann der sein, dass es eine sicherheitsrelevante Applikation ist, die nur mit einer bestimmten Version der Library geprüft und validiert wurde. Da so ein Prüflauf mit Zertifizierung aufwändig ist und daher Zeit und Geld kostet, ist es verständlich, wenn Hersteller das nicht bei jedem Update einer Library machen, zumal nicht jede neue Version einer Library dem Schließen von Sicherheitslücken dient.

Dazu kommt auch, dass es immer wieder mal vorkommt, dass eine neue Version einer Library nicht 100% rückwärtskompatibel ist und daher evtl., dass das diese nutzende Programm für die neue anzupassen ist. Dabei ist ein durchaus nicht unüblicher Fall hierbei der, dass die Programmierer sich nicht sauber an die Standards der Library gehalten haben oder sonst irgendwie unsauber programmiert haben und denen das nun um die Ohren fliegt. Ein Klassiker kann der sein, dass die alte Version der Library einen Registerwert nicht ändert, aber nicht zusagt, es unverändert zu lassen und die neue ihn ändert. Aber auch Ungereimtheiten in der Library, die das Programm brauchte und nun ausgemerzt wurden, können hier störend wirken.

Ich erinnere mich noch an ein Stück PHP-Code, bei dem eine globale Variable und der Aufrufparameter für einer Funktion denselben Namen hatten und die Funktion die globale Variable ändern sollte. Der Aufrufparameter war daher genaugenommen unnötig und Unsinn. Die alte PHP-Version hat diesen Unfug mitgemacht, weil sie die beiden an sich formal unterschiedlichen Variablen (die globale und den Funktionsparameter) nicht unterschieden hat, die neue hat es aber (was an sich auch sauber ist und den Regeln entspricht), so dass der Code plötzlich nicht mehr tat, was leicht behoben werden konnte, indem der Aufrufparameter umbenannt oder, was mehr Sinn macht, da unnötig, entsorgt wurde.

SharedLibs

Seit einiger Zeit erfreuen wir uns diverser Ports aus der Linux-Welt und damit hielt eine Library-Sammlung in Form der Applikation !SharedLibs Einzug.

Die SharedLibs entsprechen den Libraries, die man unter Linux kennt und da – wir erinnern uns – ist es kein Problem von der einen oder anderen Library mehrere unterschiedliche Versionen vorzuhalten und zu nutzen.

Unter RISC OS seint es mir so, dass es aktuell nich vorgesehen ist, dass quasi dieselbe Library in unterschiedlichen Versionen zweimal im RAM lebt. Nicht, dass das nicht möglich wäre, aber es ist fummeliger und wenn ich einen Blick in !SharedLibs werfe, sehe ich bislang von jeder Library jeweils nur eine Version.

Allerdings halte ich die Option, mehrere Versionen einer Library vorzuhalten, aktuell für eher nicht relevant, da ich davon ausgehe, dass das Gros der Software eh die neueste nimmt, sprich die Library via dem auch hier nach-

gebildeten symbolischen Link ohne Versionsangabe lädt, und somit die neueste.

Das birgt natürlich ein gewisses Risiko – vor allem, wenn bei diesen Libraries der heilige Gral der Rückwärtskompatibilität ist hier nicht so durchgängig gegeben ist oder eine neue Version noch nicht ganz fertig, was einige gerade mit der vom neuen Iris-Browser von RISC OS Developments erfahren dürfen. Bis zum offiziellen Erscheinen von Iris wird das sicherlich behoben – muss es ja auch, denn wenn der User die Wahl hat, Iris oder andere diese Library nutzende Programme zu verwenden bzw. vor dem Wechsel zwischen den Welten zu booten, ist das auf Dauer unbefriedigend.

(E)ABI

Werfe ich einen Blick in die klassischen !SharedLibs, so sehe ich im Unterverzeichnis lib neben diversen Dateien, deren Namen die Endung „so“ haben, die beiden Verzeichnisse abi-1.0 und abi-2.0 mit in diesen wieder diversen so(lchen) Dateien. In der mit Iris kommenden !SharedLibs hingegen steckt in libs ein armeabihf-Verzeichnis.

Ferner ist in der normalen lib von !SharedLibs ld-riscos/so/1, wohingegen die von Iris dort ld-riscos/so/2 enthält.

Bei genauerem Hinsehen scheinen in armeabihf all die Module drinzustecken, die ich in meiner alten shared Library drin habe, wenn auch zum Teil ein wenig in den verschiedenen Unterverzeichnissen anders verteilt.

Dabei sei angemerkt, dass „ABI“ die Abkürzung für „Application Binary Interface“ ist und entsprechend „EABI“ die Embedded-Version davon ist – also eine Schnittstelle bezeichnet. Der Suffix „HF“ steht für „Hard-Float“ und regelt, wenn ich es korrekt gefunden habe, wie Fließkommawerte übergeben werden (genauer: in den dedizierten Registern oder in normalen).

Wie dem auch sei: Zumindest für aktuelle Raspberry Pi habe ich gesehen, dass die „Arm-hard-float-Architektur (armhf)“ empfohlen wird. Da passt armeabihf schon mal gut, zumal die Architektur nicht etwa am Board hängt, sondern am Prozessor, so dass armhf auch für die anderen Rechnerchen mit nicht zu alten ARM-Prozessoren angezeigt ist.

Insgesamt habe ich den Eindruck, dass mit Iris eine neuere shared Library-Version zum Einsatz kommt und daher ein paar Unterschiede zwischen dem alten und dem neuen ABI – und das ohne, dass Pisa da eine Rolle spielt – für die Probleme gesorgt haben, dass Iris die neue aber alle anderen die alte shared Library brauchen. Dabei dürfte Iris ein paar Funktionen brauchen, die erst mit der neuen Library verfügbar sind, aber, da noch in der Implementierungsphase, das armhf-ABI noch nicht ganz final ist, weshalb Altanwendungen aktuell noch scheitern, aber das wird sicher noch gefixt.

Wenn somit eine „Nebenwirkung“ von Iris ist, dass diese zentrale Ressource auf einen moderneren Stand angehoben wird, hätte das was.

Updaten

Das Einspielen eines neuen Moduls der normalen Shared C Library ist einfach, da RISC OS-Module intern eine Versionsnummer tragen. Dazu kommt, dass sie i. a. nicht nackt, sondern als minimalistisches !System daherkommen, dass via Systemkonfigurations-Gadget (erreichbar per Doppelklick auf die Bootapplikation) eingespielt wird und dieses Tool beachtet die Modulversionsnummern.

Bei den !SharedLibs hingegen ist es leider nicht so einfach. Die Dateinamen geben zwar sauber die Version des Originals wieder, dass wir da als Portierung vorfinden, aber innerhalb der Librarydatei selbst ist keine Versionsnummer drin. Das kann zu einem Problem werden, wenn dieselbe Library erneut durch die Portierungsmühle gedreht wird, weil z. B. der erste Port irgendwo einen Bug enthielt. Andere Gründe für eine neue Version derselben Library sind Bugfixes am Compiler, sparsameres Linken, neue ARM-CPUs oder was auch immer.

Hier scheint mir, dass es das Beste ist, auf PackMan zu setzen, da die Shared Library an sich, wenn auch in mehrere Teile aufgeteilt, immerhin als Paket mit einer Versionsnummer versehen ist und somit PackMan erkennen kann, ob man historisches oder aktuelles hat und ggf. updaten sollte.

Alternativ bleibt die Option, sich auf den Timestamp der Dateien in der !SharedLibrary zu verlassen, was ein gewisses Restrisiko birgt, da es vorkommen kann, dass die Uhrzeit nicht korrekt eingestellt ist (aber dann dürfte sie mangels Uhrenchip und Zeitserver eher bei Null angefangen haben, was R-Comp mal für ein OS-Upgrade passiert ist, und somit so alt sein, dass es auffällt). Beim Saugen aus dem Internet bekommt das gesaugte Archiv je nach Art des Saugens den aktuellen Timestamp bekommt, aber innerhalb des Archivs sollte er stimmen.

Aber

Natürlich kann es auch unter RISC OS für normale Module passieren, dass der Autor es durch eine neue Compilerversion jagt, die einen Bugfix hat oder um das Modul für neuere Hardware sicher funktionstüchtig zu machen, ohne die Versionsnummer des Moduls zu ändern. Wenn das Modul dabei funktional unverändert bleibt, ist das nur ungeschickt, aber irgendwie noch vertretbar.

Doch leider hält dieses Problem nun auch anderswo unter RISC OS Einzug, wie Steffen Huber anmerkte – und das sogar in einem sicherheitsrelevanten Kontext: Dem Modul AcornSSL:

Wenn eine neue Version von mbedTLS drin steckt, ändert sich die Versionsnummer von AcornSSL nicht unbedingt (sprich es ist schon vorgekommen) und das halte ich für ein Unding, da i. a. ein neueres mbedTLS durchaus sicherheitstechnisch relevante Anpassungen mitbringt und es daher gut wäre, das dem AcornSSL-Modul auch von außen anzusehen, damit ein Einmischen via Systemkonfiguration das neue sicher einspielt und man es auch erkennt.

Handarbeit

Im Regelfall bist Du gut dabei, wenn Du normale RISC OS-Module, die in !System stecken, mit dem entsprechenden Gadget der Systemkonfiguration einspielst. Für die !SharedLibs hingegen rate ich aus den oben genannten Gründen, PackMan zu verwenden.

Wer aber absolut auf Nummer Sicher gehen will und den o. g. Tools nicht vertraut, sollte genau wissen, was er tut, braucht eine ausgefeilte Backupstrategie und Geduld. Mein Tipp ist, da einfach mal der Systemkonfiguration für normale Module und PackMan für das unixoide Zeug zu vertrauen. ❍

PLPP

PHP – Lua – Perl – Python

Herbert zur Nedden

Es buhlen ein paar interpretierte Sprachen neben dem klassischen BBC Basic sowie dem neuen DARIC (ersteres kennst Du, zu letzterem findest Du ein paar Informationen an anderer Stelle in dieser News) um die Gunst der RISC OS-User, wobei Python 3 insofern besonders hervorsticht, als es die Chance hat, eher neue User nach RISC OS zu holen, da es eine dieser Zeit verbreitete „Einstiegsdroge“ für die Programmierung ist und gerade recht aktuell aktualisiert wurde.

Da musste ich mal einen kurzen Blick riskieren:

PHP

PHP ist ein rekursives Akronym und Backronym für *„PHP: Hypertext Preprocessor“* (also ähnlich wie GNU, dass für *„GNU's not Unix“* steht); ursprünglich stand PHP für *„Personal Home Page Tools“*. Es ist eine an C und Perl angelehnte Skriptsprache, die primär zur Erstellung dynamischer Webseiten oder Webanwendungen verwendet wird. Sie auch dank vieler Funktionsbibliotheken und der Unterstützung diverser Datenbanken recht mächtig.

Nach einem Download von PHP zu googeln ist relativ sinnarm, denn suche ich nach „RISC OS PHP“ (ob nun „RISC OS“ in Gänsefüßchen eingerahmt, damit es als ein Wort gilt, oder nicht), sind fast alle Fundstellen uninteressant, weil der String „PHP“ in diversen URLs vorkommt. Der Trick ist, nach folgendem zu suchen, sprich zwar nach „PHP“ aber explizit nicht nach „.php“:

```
"RISC OS" PHP -".php"
```

Die Heimat von PHP selbst ist www.php.net und dort findet man PHP v8, wobei das noch recht frisch ist, da von November 2020 stammend – daher dürfte v7 deutlich verbreiteter in freier Wildbahn zu finden sein und wird ebenfalls noch aktiv gepflegt.

Die Suche nach „RISC OS“ auf www.php.net liefert lediglich die Information, dass da nix dazu zu finden ist, so dass wir als Kenner der Szene direkt zu www.cp15.org/php wandern, wo PHP für RISC OS zu finden ist (nebst WebJames übrigens).

Entpackt bekommen wir ein gutes Dutzend Dateien, die bummelig 8 MB auf der Platte belegen und eine StrongHelp-Datei sowie, leicht irritierend die php.ini-Datei nicht in, sondern neben der PHP-Applikation. Die liegt separat, da sie wahrscheinlich beim Einsatz in einem Webserver diesem vorgelegt werden darf.

Die Dokumentation ist nicht RISC OS-Spezifisch – vielmehr scheint es eine ins StrongHelp-Format gebrachte Standarddokumentation zu sein, da bei der Installationsanleitung alles Mögliche außer RISC OS Erwähnung findet.

Beim Start der PHP-Applikation folgt die Frage, ob man es standalone – wahlweise single- oder multitaskend – wünscht, oder als CGI in ANT, Navaho, Netplex oder WebJames; hier fehlt natürlich der Webserver HTTPServ, aber das liegt vielleicht einfach am Alter des Ports. An dieser Stelle ist der Help-Button gut, da er mit HTML-basierten Hilfe auch weiterhilft, denn bei Netplex denken viele dieser Tage vielleicht an einen Schreibfehler (Netflix) und Navaho ist wohl auch wenn, dann nur wenigen bekannt.

Aktuell würde ich PHP, wenn als CGI, in WebJames nutzen, da beide vom selben Autor betan werden, was vermuten lässt, dass die beiden auch gut harmonieren. Alternativ wäre da noch der HTTP-Server HTTPServ von Thomas Milius ein denkbarer Wirt, da der zumindest von Thomas aktiv genutzt wird und nicht zu historisch ist. Aber das zu beäugen, hebe ich mir für eine spätere Ausgabe der News auf…

Für den Standalonebetrieb, den ich für die erste Inaugenscheinnahme wählte, ist in der Hilfe noch vermerkt, dass es bei den meisten Skripten keinen nennenswerten Unterschied machen würde, ob man PHP single- oder multitaskend einsetzt – sprich die Laufzeit ist wohl i. a. eher kurz. Und natürlich ist auch erwähnt, dass die Standaloneversion, sprich die ohne Webserver dahinter, Dinge wie GET und POST die man für Formulare nutzt, keine Chance haben, was aber auch klar ist, da die einen Webserver für die Interaktion brauchen.

Also kurz mal eine Textdatei mit diesem Inhalt angelegt:

```
<html><body>
<?php echo 'Hallo PHP ' . phpversion(); ?>
</body></html>
```

…und mit dem Dateityp PHP (&18A) gesichert und doppelt angeklickt.

Kurz darauf erscheint im Browser die Ausgabe „Hallo PHP 5.2.2“ (war der Browser nicht geladen, wird er dafür geladen). Die Version 5.2.2 ist insofern eine Überraschung, weil der PHP-Download unter der Version 2.23 firmiert.

Was aus der Ausgabe oben deutlich wird, ist, dass das für RISC OS verfügbare PHP aus dem Jahre 2009 stammt und somit schon recht museal ist. An sich ist alte Software normalerweise kein Thema, sofern sie das kann, was man damit tun will, aber in diesem Fall kommt dazu, dass sie dafür gedacht ist, einen Webserver funktional aufzupeppen und ein so altes PHP würde ich eher nicht im Internet anbieten wollen – wobei es sein kann, dass es schon wieder so alt ist, dass es dadurch „sicher“ ist. Aber in reinen Intranet-Anwendungen kann man die Version natürlich gerne nutzen.

PHP ist, wie sicher bekannt, eine Sprache für Webseiten etc. und daher ist es auch passend, dass PHP für die Ausgabe nicht etwa einfach ein Taskfenster o. ä. bemüht, sondern den Browser. Als quasi „normale“ Skriptsprache ist es eh eher nicht gedacht – auf der anderen Seite ist es praktisch, dass man dafür recht einfach eine grafische Oberfläche bieten kann, indem man die Ausgabe mit ein wenig HTML-Code anhübscht und dann den Browser den Rest machen lässt, was auch ältere RISC OS-Browser locker schaffen, sofern Du nicht zusätzlich mit JavaScript etc. operierst.

Insgesamt kannst Du somit mit WebJames oder HTTPServ und PHP durchaus Anwendungen schreiben und Dir dabei in Bezug auf die grafische Oberfläche das Leben insofern recht einfach machen, da Du Dich nicht mit Wimp-SWIs herumschlagen musst, sondern lediglich ein wenig HTML-Code absonderst. Das mag zwar nicht so flexibel sein, wie echte RISC OS-

Applikationen, aber für diverse Dinge dürfte es schon genügen und nebenbei sind solche Anwendungen auch locker client-server-tauglich.

Wie angedeutet, werde ich PHP mit einem Webserver demnächst mal in Augenschein nehmen.

LUA

Lua ist portugiesisch für Mond, was sich im Logo der Sprache widerspiegelt. Man kann in Lua eigenständige Programme schreiben, aber diese Sprache ist eher als eingebettete Skriptsprache für andere Programme konzipiert und ist kompakt und schnell.

Auf der Heimatseite www.lua.org findest Du Version 5.4.2 von Ende 2020 und der RISC OS-Port, der von www.wra1th.plus.com/lua/risclua.html saugbar ist, basiert auf Version 5.4.2, also der aktuellen – aber auch hier hat der Portierer eine etwas eigenwillige Versionsangabe unter RISC OS, da das Risc Lua ausgepackt als Applikation !rlua84 daherkommt, aber immerhin ist im Namen des Downloads die 5.4.2 enthalten.

Lua gibt es in zwei Versionen: Mit und ohne VFP – sprich für aktuelle und ältere Hardware. Ausgepackt fand ich 150 Dateien vor, die gut 1MB belegen und ein StrongHelp ist auch zur Hand.

Erfreulich ist, dass die Hilfe erläutert, wie du SWIs aus Lua heraus rufen kannst, so dass Du mit diesem Lua-Port an sich alles unter RISC OS machen kannst, wonach Dir der Sinn steht.

Was dort auch zu lesen ist, ist dass es eine Variable namens „version“ gibt, die verrät, welche Lua-Version da am Start ist – aber die Variable ist leer. Daher legte ich zwar wieder eine Textatei an, aber dieses Mal mit nur diesem Inhalt, sprich ohne Ausgabe der Versionsnummer:

```
Print ('Hallo LUA')
```

Dateityp Lua (&18C) und schon ist sie per Doppelklick ausführbar.

An sich macht dieser Lua-Port eine recht brauchbare Figur, ist gut nutzbar und soweit mir bekannt, ist die eine oder andere Wimp-Applikation in Lua geschrieben worden. Und der Port basiert auf der aktuellen Lua-Version, was löblich ist.

Allerdings muss ich zugeben, dass ich damit nicht ernsthaft arbeiten werde, da ich als normale Programmiersprache C präferiere (oder für Kleinigkeiten auch mal Obey-Dateien oder Basic). Zugegeben, eine gute Skriptsprache, die die Kommandozeile aufwertet und auch Funktionen bietet, ist auch gerne genommen, aber da passt Lua nicht ganz in mein Beuteschema.

Perl

Perl entstand ursprünglich als Werkzeug zur Verarbeitung von Textdateien in der unixoiden Welt (was man auch an der Syntax der ersten Zeile eines Perlskripts sieht) und wurde u. a. durch C und awk inspiriert. So diente es u. a. zur Auswertung von Logdateien, wird aber auch bei Webanwendungen, in der Bioinformatik und Finanzwelt genutzt, um z. B. Datenströme aus verschiedenen Quellen zu aggregieren. Es ist somit nicht überraschend, dass Perl vor allem bei der Verarbeitung von Texten mit regulären Ausdrücken sehr mächtig ist.

Perl's Heimat ist www.perl.org und da finden wir aktuell Version 5.32.1 und, wie schon befürchtet, keinerlei Spuren von RISC OS. In RISC OS 5 selbst ist auch ein wenig Perl drin, aber wie im Forum von RISC OS Open zu lesen, älter und unvollständig und meldet sich als Perl 5.001 aus dem letzten Jahrhundert.

Ich vermute mal, die Perl-Version 5.8.8 von September 2006 von www.cp15.org/perl lohnt die Mühe des Downloads nicht, denn auf www.riscosports.co.uk/downloads.html finden wir immerhin Perl 5.14.2 und somit ein nicht so arg altes, aber die 5.14er-Perle stammt aus 2011 und ist somit eine Dekade alt – aber immerhin haben wir schon mal die aktuelle Hauptversion.

Ausgepackt erhalten wir gut 2600 Dateien, die knapp 45 MB belegen, wobei ca. ein Viertel davon alleine auf die im HTML-Format beigefügte Dokumentation entfällt.

Kurz eine Textdatei folgenden Inhalts mit Dateityp Perl (&102) angelegt:

```
#!/user/bin/perl
print "Hallo PERL $^V\n";
```

Nach dem Doppelklick erscheint dann auch die Versionsnummer 5.4.12 auf dem Schirm.

Insgesamt sicherlich für einige Dinge eine gute, hilfreiche Sprache – schade nur, dass da gerade drei verschiedene und obendrein durchweg alte bis sehr alte Versionen für RISC OS herumschwirren. Doch gilt es, Textdateien, Logs oder was auch immer auf dem lokalen Rechner durchzuflöhen, dann ist es nicht weiter schlimm, dass das Perl für RISC OS ein wenig älter ist – Hauptsache, es ist der ihm gestellten Aufgabe gewachsen.

Python

Python weicht von den drei obigen insofern ein wenig ab, weil es eine höhere Programmiersprache ist – passt aber schon irgendwie in diesen Artikel, da in der Regel interpretiert (wie Basic auch).

Anders als viele andere Sprachen nutzt Python nicht die gerne genommen (meist geschweiften) Klammern, sondern Einrückungen zu Strukturierung – wehe dem, der dabei ein Mischmasch von Tabulatoren und Leerzeichen verwendet, da Texteditoren Tabs nicht einheitlich darstellen (gängig sind Vierer- und Achterschritte), was schlimmstenfalls irritiert.

Python unterstützt mehrere Programmierparadigmen wie die objektorientierte, die aspektorientierte und die funktionale Programmierung und rundet das mit dynamischer Typisierung ab.

Auf www.python.org ist die Heimat von Python und dort ist die 3.9er-Version von Ende 2020 die aktuelle. Für RISC OS haben wir Version 3.8 als reguläres Paket zur Hand, also schon recht up to date. Die 3.8er stammt zwar von Ende 2019, wird aber noch ein paar Jahre supportet.

Bemerkenswert und erfreulich ist, dass auf der amtlichen Webseite von Python auch erwähnt wird, dass es Python für RISC OS gibt und man es dafür mit dem Paketmanager PackMan installieren kann.

Hier wählte ich den empfohlenen Ansatz und bemühte PackMan für die Installation (alternativ hätte ich auch nachsehen können, von wo PackMan die Sachen saugt und dann eine händische Installation machen können).

Wichtig ist, erst einmal python-core zu installieren, dass quasi der Überbau mit den drei Applikationen !Python3 (da kommt gleich Python selbst rein), !PythonSite und !PythonUser ist.

!Python3 selbst will PackMan nach Apps.Development packen, was an sich noch ok ist. !PythonSite scheint ein paar Ressourcen zu enthalten, da es in !Boot.Resources kommmen soll – soweit immer noch gut. Dass !PythonUser in PreDesk stehen soll, verwunderte schon. In PackMan kannst Du das Installationsziel der drei auch ändern, was gut ist, da ich auf meiner Platte keinen Apps-Ordner im Root habe – aber für das Gros der User, die nicht, wie ich es tat, die Verzeichnisstruktur auf der Platte umgekrempelt haben, sind die vorgeschlagenen Pfade schon gut.

Als zweites Paket holen wir uns python selbst – der Download dauert länger, da nun auch richtig etwas gesaugt wird und danach belegen die drei o. g. Applikationen zusammen mit knapp 2300 Dateien rund 50MB auf der Platte, wovon die Library, die in !Python3 landet, das Gros des Platzes einnimmt.

Nun habe ich mal einen genaueren Blick auf die drei Komponenten geworfen. !Python3 ist vital, weil hier Python selbst drin steckt und auch der Unterordner Scripts mit ein paar Skripten und der Ordner Python38 mit einer Library, passend zum installierten Python 3.8. Das Ausführbare Python ist übrigens im ELF-Format – aber dass wir die !SharedUnixLib etc. brauchen, war ja schon bei der Installation in PackMan deutlich.

Da mein RISC OS an sich recht aktuell ist, was die Shared Libraries betrifft, musste ich dafür übrigens nicht Hand anlegen – aber bei der Gelegenheit mal einen Blick zu werfen, ob sie noch aktuell sind, kann nie schaden.

Zu Python gehören üblicherweise je ein Container für die von der Site abhängigen Dinge sowie die der User – soweit passt es mit !PythonUser und !PythonSite. Beide fügen einen in ihnen enthaltenen Script-Ordner in den Run$Path ein, wobei beide diesen Ordner im Auslieferungszustand nicht enthalten. Und warum !PythonUser vor dem Starten des Desktops gebootet werden soll, ist mir unklar, denn ohne, dass Python selbst startklar ist – und das wird es erst später, da in Apps beheimatet – ist der Nutzen doch arg limitiert; konzeptionell hätte ich !PythonUser eher in Choices erwartet. An sich kannst Du daher !PythonUser und !PythonSite sonstwo ablegen, wobei !Boot.Resources wahrscheinlich für beide ein guter Platz ist, zumal sie dann beim Rechnerstart gebootet werden.

Wieder gibt es eine Textdatei, diesmal mit diesem Inhalt:

```
import sys
print("Hallo PYTHON "+sys.version)
```

Der Dateityp ist nun Python3 (&A73) und der Doppelklick führt sie aus und meldet dann auch, dass Python in der Version 3.8 am Start ist.

Dazu kommt als Schmankerl, dass man aus Python3 heraus auch SWIs aufrufen kann, wie diese Aufrufe zeigen:

```
import swi
swi.swi('OS_NewLine','')
swi.swi('OS_Write0','s','Hello again!')
```

Damit kann man mit Python3 unter RISC OS lecker schalten und walten.

Python

Von packages.lessthan3.org.uk/pkg/Python-310-3.10.0-1.zip kannst Du Python 3.10 saugen – auf der Webseite von Python ist die Version 3.10 übrigens bei den Pre-Releases zu finden, sprich ist quasi überaktuell.

Bitte beachte, dass das nur das „nackte“ Python beinhaltet, also nicht die !Python3-Applikation mit dem Überbau, der es mittels Systemvariablen bequem nutzbar macht.

Zum Testen habe ich aus der 3.8er !Python3-Applikation die paar Klein- ode außer den Unterverzeichnissen in die 3.10er kopiert – alternativ hätte ich auch das frisch Gesaugte in die alte Applikation stopfen können, da sie automatisch erkennt, welches das neueste Python da drin ist und das auch nutzt.

Die ersten Gehversuche von Python 3.10 endeten beim Aufruf von help() in einer ein wenig kryptischen Zeilensammlung (solche sogenannten Backtraces sind für normale User eher irritierend, für den versierten Programmierer durchaus hilfreich), von der die letzte Zeile klar verständlich war:

```
swi.error:
   File 'System:Modules.TaskRunner'
   not found
```

Es fehlt das Modul TaskRunner – stimmt, das hatte ich nicht auf meinem System. Also kurzerhand PackMan bemüht, das Modul dort gesucht und gefunden; und kaum installiert, funktioniert der Aufruf von help() fehlerfrei. Wenn das neue Python in PackMan landet, dann wird dich PackMan natürlich auch darauf hinweisen, dass TaskRunner benötigt wird.

Mein kleines Script aus der Textdatei tut auch, wenn ich die Kommandos direkt auf der Python-Kommandozeile eintippe, aber wenn die meine kleine Python-Datei doppelt anklicke, erfahre ich, dass die Datei nicht zu finden ist – kein Wunder, wird doch der Datei selbst noch total überflüssig der Pfad zum Root des Bootlaufwerks vorangestellt; ich denke, das ist Kleinkram, der bald vom Tisch ist, denn der 3.10er-Port ist ja gerade erst im Werden.

Bilanz

PHP ist alt, richtig alt – aber ich denke, wer ernsthaft einen Webserver betreibt und diesen ins Internet hängen will, dürfte in der Regel eh zu Linux greifen – also z. B. einem debian auf dem Pi, da dafür jedwede Securityfixes sehr schnell am Start sind – oder einfach den Service seines Providers dafür nutzen und so das Aktuellhalten, so in qualifizierte Hände legen.

Aber für die eine oder andere lokale Applikation ist PHP schon bequem, da Du einfach ein wenig HTML ausgeben musst und ein hübsches Fenster zu Gesicht bekommen kannst, ohne, dass Du dich mit all den Wimp-SWIs herumquälen musst. Und mit WebJames oder HTTPServ ist es auch möglich, Dialoganwendungen zu stricken und die sogar im Intranet von mehreren Rechnern aus zu nutzen.

Auch kann man das PHP sicherlich (wenn auch eventuell mit einzelnen Einschränkung ob der Version) unter einem Webserver unter RISC OS als Entwicklungsumgebung für eine Webanwendung nutzen, die dann später auf einem Server im Internet zum Einsatz kommen soll.

Der Lua-Port ist wirklich aktuell, aber irgendwie stolpere ich im normalen Leben eher nicht über diese Sprache und konnte mich bislang auch nicht für Lua erwärmen – das ist kein Makel der Programmiersprache, sondern liegt daran, dass ich gerade mit C, Basic und Obey alles habe, was ich unter RISC OS so brauche… naja fast, denn hie und da hätte ich schon gerne so eine einfache Skriptsprache, mit der man bequem Kommandozeilenskripte schreiben kann, denn da bin ich von der Bash unter Linux und der PowerShell von Windows arg verwöhnt.

Bei Perl ist es ein wenig bedauerlich, dass die in RISC OS 5 enthaltenen Teile so museal sind und auch die verfügbaren Ports schon mehr oder weniger angestaubt sind. Auf der anderen Seite arbeitet es ja eher lokal und da ist das Alter zumindest aus Sicherheitsgründen unter RISC OS wohl eher Nebensache. Und mal eben Textdateien zu durchflöhen, dürfte die vorhandene Version schon sehr hilfreich sein und bei ernsthaftem Bedarf, den ich nicht habe, eventuell schon eleganter, als sich mit awk und sed abzumühen oder mal eben ein eigenes Analysetool zu stricken.

Wenn ich bislang mal mit vielen Textdateien zu tun hatte, waren das eigentlich zur zwei Fälle: Entweder die C-Quellcodedateien der GAG-C-Library oder größere Mengen von E-Mails. Meist wollte ich dabei nur bestimmte Dinge finden oder in seltenen Fällen mal globale Änderungen machen. Für beides genügte mir StrongED – mich dafür in Perl oder etwas anderes einzulesen hätte sich auch nicht gelohnt, da der Bedarf zu selten aufkam. Aber bei häufigerem Bedarf wie z. B. dem regelmäßigen Auswerten von Serverlogs, um zu sehen, ob da Dinge passieren, dürfte Perl gut geeignet sein, um da einiges zu automatisieren.

Bei Python ist es definitiv gut, dass das in recht aktueller Version für RISC OS verfügbar ist und sogar die kommende Version schon in Vorbereitung – und auch, dass auf der offiziellen Webseite von Python zu lesen ist, dass es für RISC OS verfügbar ist.

Dazu kommt, dass wohl Python gerade für Einsteiger derzeit eine gerne genommene Programmiersprache ist, die auf diversen Plattformen verfügbar ist. Damit dürfte das Vorhandensein von Python für RISC OS helfen, dass sich neue User eher mal mit RISC OS befassen, zumal ein Raspberry Pi unter RISC OS von der Performance her Raspbian irgendwie alt aussehen lässt. Zugegeben, der erste Aufruf von Python dauert einen Moment, aber dann sind ein paar MB RAM belegt (aber RAM haben die aktuellen RISC OS-Rechner ja im Überfluss) und Python recht reaktionsfreudig.

Und da man aus Python3 heraus auch SWIs aufrufen kann, kann man sich mit dieser Sprache so richtig unter RISC OS austoben.

Interpretiert?

Zugegeben, eine Interpretersprache mutet zumindest für sogenannte-oder-auch „gestandene“ Programmierer ein wenig befremdlich an, da solche Sprachen systembedingt tendenziell nicht so performant sind, wie Kompilate und obendrein der Quellcode einsehbar ist – aber auf der anderen Seite findet man massenweise gute in Basic geschriebene Software für RISC OS und somit in eben so einer Sprache und für viele Arbeiten dienen Batchdateien, die es auf fast allen Plattformen gibt und auch die werden interpretiert. Ferner gibt es mittlerweile diverse Sprachen auf halbem Wege dazwischen, da sie beim Start per Just-in-Time-Compiler oder so ähnlich betan werden, um dann zu performen. Und die immer noch laufend steigende Performance der Rechner selbst, die wir seit einiger Zeit sogar unter RISC OS erleben dürfen, relativiert die relative Trägheit der Interpretation zusätzlich.

Auch ist es für Einsteiger einfacher, mal ein wenig zu codieren und direkt auszuprobieren – oder gar mal das eine oder andere Kommando direkt im Interpreter abzusetzen, um zu sehen, was es tut bzw. in der Fehlermeldung zu erfahren, wo man sich vertippt hat.

Wenn ich mich erst in eine Entwicklungsumgebung einarbeiten darf und vor jedem Test der Anwendung Compiler und ggf. Linker anwerfen, ist das für mich das normalste von der Welt, aber für jemanden, der gerade die ersten Gehversuche macht, eher ein Buch mit sieben Siegeln, sprich eine unnötige Komplexität – und das selbst, wenn die Entwicklungsumgebung einem das Leben da durchaus relativ einfach macht und mit einem Debugger die Fehlersuche erleichtert.

Langfristig ist natürlich für größere Projekte eine umfassende Entwicklungsumgebung mit eingebauter Dokumentation der Sprache und eventueller Libraries, einem Editor mit Syntaxhighlighting und interaktiver Hilfe, Debugger und natürlich Compiler, Linker etc. sicherlich sehr hilfreich – etwas, wo wir unter RISC OS, sagen wir mal so, viel Potential haben, denn das DDE bietet dafür zwar irgendwie das meiste, ist aber in der Hinsicht schon recht rudimentär.

Und…

Da hätte ich doch fast noch eine Sprache vergessen, die bei uns recht verbreitet ist und gerne interpretiert wird: „Deutsch“. ❍

Wer lesen kann…

!MuView ist besser

Herbert zur Nedden

MuView erlaubt doch das Ändern der Skalierung der Seiten und sogar noch mehr – wobei man, um das zu lernen, die Hilfe lesen muss, da die Tastaturcodes dafür untypisch sind und die Option weder als Buttons noch im Menü verankert sind.

Dass Blättern mit den Bild Hoch/Runter-Tasten klappt, war klar und ist auch normal. Nur zum Vergrößern oder Verkleinern dienen nicht die Plus- und Minus-Tasten, sondern die Pfeiltasten nach oben und unten; die Rechts- und Linkspfeile hingegen drehen die Anzeige. Via Menü schließlich – aber das findet man ja ohne Literaturstudium – ist jede Seite direkt ansteuerbar.

Vielleicht hätte ich doch die Hilfe genau durchlesen sollen, statt bei so einer Applikation einfach mal das übliche zu vermuten… Danke für den Hinweis, Raik. ❍

And the winner might be...

RISC OS Awards 2020

Herbert zur Nedden

Im April ging auf riscosawards.co.uk/2020 die Abstimmung für die RISC OS Awards 2020 online – dem voraus ging eine Diskussion im Forum von RISC OS Open, wie unter Kurz Notiert zu lesen. Genaugenommen deckt die Abstimmung dabei sogar 14 Monate ab, da sie aus nicht näher erläuterten Gründen den Zeitraum ab November 2019 abdeckt.

Wie üblich sind in fast allen Kategorien ein halbes Dutzend Kandidaten nominiert und wenn Du einen besseren Vorschlag hast, kannst Du den über das Textfeld einbringen, wobei diese Möglichkeit bislang leidglich den Nutzen hatte, dass man eher sein Gewissen beruhigen konnte, indem man für das stimmte, was man für das beste hielt, aber gewonnen hat bislang nie ein so ergänzter Kandidat, aber immerhin wurden häufig hier genannten dann gelegentlich zumindest mal auf dem Twitter-Feed von RISCOSitory als Anregung für andere erwähnt.

Alternativ kannst Du keine der angebotenen Optionen wählen und in das Feld für eine Alternative nur einen Kommentar für den Veranstalter eintragen, denn ohne, dass Du explizit die „Alternative option“ selektierst, wird dieses Feld nicht als Wahlentscheidung, sondern als Anmerkung interpretiert.

Selbstmurmelnd kannst Du auch beschließen, keinem der Kandidaten Deine Stimme zu geben – soweit noch ok. Aber das Feld als „No opinion“ zu bezeichnen, sprich mit einem Text, der als „keine Meinung“ übersetzt werden kann, ist sicherlich auf Anhieb irritierend, denn wenn ich keinem der Angebotenen meine Stimme geben will, besagt das sicher nicht, dass man keine Meinung hat, sondern es kann ja auch sein, dass man sich nicht entscheiden kann, weil zwei gleich gut sind, oder keinen der Kandidaten für würdig erachtet. In diesem Zusammenhang sei noch zu „Opinion“ angemerkt, dass – *„He holds no opinion“* (beachte, dass ich *„holds“* und nicht *„has“* schrieb) aus als *„Er ist neutral“* übersetzt werden kann und meiner Meinung nach auch sollte.

Und wie üblich ist bei der Frage nach der oder dem, der den größten/besten Beitrag für RISC OS geleistet hat, keine „Hilfestellung“ in Form einer Vorauswahl im Angebot – an sich eine gute Idee, nur, dass die, die eher diskret im Hintergrund werkeln so leicht schlechtere Gewinnchancen haben, aber das hat ja bislang auch recht gut geklappt.

Doch gibt es noch eine zweite Nominierung, die dieses Mal nur ein Freitextfeld bietet: Die beste Show oder das beste Event, da im Zeitraum, für den diese Ehrungssuche läuft, nur die Southwest Show 2020 im realen Leben stattfand, die London Show hingegen online und obendrein die Usergruppentreffen ob der Tatsache, dass auch die virtuell stattfinden, mehr Zulauf bekommen haben.

Kandidaten

Hier nun die Kandidaten mit dann den üblichen Kommentaren meinerseits. Bei den Überschriften erspare ich mir das „Beste“ o. ä., da das für alle gilt und – anders als auf der Webseite – hier der Platz ein wenig limitierter ist und ich so mit einzeiligen Überschriften auskomme.

Achtung
Früher gab es „normale“ Software und „Games“ – mittlerweile findest Du ein paar weitere dedizierte Software-Kategorien, aber leider hat der Macher der Umfrage die beiden globalen eher kommentarlos vor die speziellen gepackt. Also bitte beim ersten Blick nicht wundern, dass urplötzlich Dinge wie die Entwicklungsumgebung DDE oder der Browser NetSurf bei kommerzieller bzw. nichtkommerzieller Software nicht nominiert wurden.

Kommerzielle Software

- Fireworkz Pro – R-Comp
- RiscOSM – Sine Nomine
- Organizer – North One Communications
- TextEase Studio – Elesar Ltd
- CDVDBurn – hubersn Software
- DeskWatcher – Thomas Milius

Gutes Timing Thomas, denn DeskWatcher ist im Dezember 2020 erschienen und hat es daher (wenn auch zeitlich knapp) in diese Liste geschafft. Dass CDVDBurn hier mit im Sortiment sein würde, war klar.

Gut erkennbar ist, dass hier nur Softwaren genannt werden, für die es im Untersuchungszeitraum Neuerungen gab, denn anders ist nicht zu erklären, dass Programme wie ArtWorks oder TechWriter, die nach meinem Dafürhalten deutlich wertiger als ein paar der anderen in obiger Liste sind, fehlen. Bei Impression Publisher denke ich, ist der Umstand, dass es um die neue 32bittige Version in der letzten Zeit sehr ruhig war, Grund dafür, dass dieses Kleinod hier nicht mitspielt.

Für mich war die Entscheidung zu zwei Dritteln einfach – auch in Anbetracht der offensichtlichen Kriterien, dass es Neues gegeben haben muss, da ich somit eben die Menge des Neuen auch einfließen ließ, womit Fireworkz und RiscOSM sowie Organzier schon mal hinter den anderen dreien landeten. TextEase ist ja ganz nett, aber – überspitzt formuliert – wird da gerade gefühlt eine Sammlung einfacher Tools mit relativ viel Tamtam propagiert.

So, dass soll hier genügen, denn ich will Dich ja nicht über die Gebühr beeinflussen :-)

Nichtkommerzielle Software

- KPDFUtil – Kevin Wells
- Launcher – Steve Fryatt
- AppUtils – Steve Drain
- DplngScan – Chris Johnson (David Pilling)
- KinoAmp – André Timmermans
- MIDIUSB – Dave Higton and Rick Murray

Also ein Tool, um PDFs zu verarbeiten, ein Applikationslauncher, Tools um Resources. $.Apps zu betun, eine Bitmapbildverarbeitung, die scannen kann, ein Filmabspieler sowie MIDI-Software – das ist schon abwechslungsreich und ob des Umstandes, dass einige Softwaretypen nicht mehr in dieser allgemeinen Rubrik stehen, ist die Auswahl hier breiter und die Wahl schwieriger.

Hier kann ich beim besten Willen keine wirkliche Empfehlung geben, da es sehr davon abhängt, was Du an sich brauchst und machst. Die Hälfte der obigen sind für mich an sich von gar

keinem Interesse als Nutzer und die anderen drei fallen in die Rubrik von wenn, dann eher selten genutzter Dinge.

Game oder Ablenkung

- SCUBA Hunter – AMCOG Games
- Virus – Richard Murray
- Quizzics – Gareth Lock
- ScummVM – Cameron Cawley (RISC OS-Port)
- Stargate – Terry Swanborough
- Gorillas – David Williams

Wie üblich schreibe ich zu dieser Rubrik mit mehr oder weniger anspruchsvoller und grafisch ansprechender Zeitvertreibsoftware nichts. Ich spiele zwar gelegentlich unter RISC OS aber dann eher mit Spielen, die schon älter sind (und da gibt es auch durchaus Titel, die mehr fürs Auge bieten, als ein paar der neuen Games).

Internet/Netzwerk-Software

- Messenger Pro – R-Comp
- Pluto – Martin Avison and Rob Sprowson (ursprünglich Jonathan Duddington)
- NetSurf – The NetSurf Developers
- Chatcube – RISC OS Cloverleaf
- Sargasso – Chris Gransden (James Bursa)
- FTPc – Colin Granville

… also die Software rund ums Internet.

Sargasso ist ein Tool für RSS-Feeds und somit funktional ein Exot, FTPc hat TLS gelernt, bietet aber leider nur das normale FTP aber kein unixoides sftp. Pluto ist als Mailclient, da kein IMAP bietend, eher uninteressant; was ich von Messenger Pro halte, der nur ein TLS bietet, dass offiziell abgelöst wurde und seit Jahren ungefixte Schwächen hat, ist hinlänglich bekannt. NetSurf wird laufend weiterentwickelt und dürfte, bis Iris am Start ist, der beste Browser für RISC OS bleiben.

ChatCube ist der erste, durchaus nette Chatclient für RISC OS, der leider aktuell nur sein eigenes Format und Telegram bietet; ersteres ist eher sinnarm, da kaum User und nicht auf Smartphones, letzteres ist für mich aus Sicherheitsgründen ein No-Go; aber Besserung ist hier in Sicht.

Alles in allem also in gewisser Form suboptimal, was an dieser Front genannt wurde, zumal hier auch LanManFS/98 sowie Sun/MoonFish gut reingepasst hätten, die sich von obigen insofern unterscheiden, dass sie sich im Alltag durchaus gut bewährt haben – aber das halt ohne Updates tun.

Hier wären meine Favoriten ChatCube, da damit etwas definitiv Neues für RISC OS am Start ist (und wenn das Crowdfunding klappt und das Teil Signal lernt, wird es interessant), und NetSurf als Browser, die laufend besser wird, zumal Iris noch nicht da ist.

Entwicklungstool

- Desktop Development Environment 30 – RISC OS Open Ltd
- Gnu Compiler Collection (GCC) – GCCSDK-Developer
- WinED – Steve Fryatt
- Python – Chris Johns
- RiscLua – Gavin Wraith
- Git Client – Kevin Swinton

Bei dieser Rubrik ist das Angebot ein wenig „unfair“, finde ich, da zwei umfangreiche Compilerpakete (die ersten beiden) gegen einen Templateditor, einen Client für Git sowie zwei Sprachen antreten.

WinED würde ich hier nicht wählen, da das einfach das „kleinste Licht“ hier ist; RiscLua kommt, obwohl hochaktuell bei mir auch nicht durch, weil es gegen Python, dass auch ob des Umstandes, dass es sich an sich großen Interesses erfreut und so User zu RISC OS locken kann und ebenfalls recht aktuell ist und obendrein die kommende Version schon in Arbeit, keine Chance hat.

Der Git Client ist insofern für RISC OS durchaus wichtig, da RISC OS Open auf Git als Repository setzt und es Entwicklern ermöglicht, damit unter RISC OS zu arbeiten.

Bei den beiden Compilersuiten fällt die Wahl ein wenig schwer, denn DDE ist performanter und der Haus-und-Hof-Compiler für RISC OS, kostet aber Geld, wohingegen GCC speziell für Portierungen der Compiler der Wahl ist und somit ebenfalls ein klares Must-Have für RISC OS.

Irgendwie tendiere ich zum Git Client und Python, da ersteres neu und für RISC OS wichtig ist und letzteres nicht nur aktuell, sondern auch Potential für neue User bietet – was die beiden Compilersuites nicht abwerten soll; zumal beide eine laufende Pflege erfahren (was wichtig ist).

Hardware

- Raspberry Pi 400 – Raspberry Pi Foundation
- 4té – R-Comp & Wi-Fi Sheep
- RaspberryRO 4 – CJE Micro’s
- FOURtress – RISCOSbits
- WiFi+RTC HAT – Elesar Ltd

Vier Rechner, die auf dem Raspberry Pi 4 basieren und eine Erweiterung, deren primäre Existenzberechtigung daher kommt, dass es keinen Treiber für den WLAN-Chip auf dem Raspberry Pi gibt.

Der Pi 400 setzt sich ein wenig von den anderen dadurch ab, dass er eine Tastatur (kleiner Nuancierung) bietet und preislich nicht in den in UK üblichen Regionen für Pi-basierte RISC OS-Rechner liegt. An sich sind die fertigen Rechner schon praktisch, da man halt ein Komplettpaket bekommt, dass gleich tut und bei dem alles in einem Gehäuse steckt – welcher Dir dabei am meisten zusagt und wo das Preisleistungsverhältnis für Dich akzeptabel ist, vermag ich nicht zu beurteilen.

Ich bleibe jedenfalls vorerst beim mini.m mit eSATA-Platte. Sobald aber ein Raspberry Pi 4 mit USB 3 oder gar SATA am Start ist und somit einer netteren Festplattenperformance, komme ich sicher ins Grübeln… sofern dann nicht der Pi 5 da ist und ich warten muss, bis RISC OS da drauf tut…

Damit ist sicherlich klar, dass ich für diese Rubrik keinen wirklichen Tipp geben kann – oder vielleich doch: Lies mal den Wakefield-Showbericht und dort den Abschnitt von RISCOSbits.

Rückwärtskompatibilität

- ADFFS – Jon Abbott
- Aemulor – Adrian Lees
- ArcEm – Various
- ArchieEmu – Jan de Boer
- RPCEmu – Matthew und Peter Howkins
- VirtualRiscPC – VirtualAcorn

OK, die alten Bekannten: Der Emulator, mit dem viele, viele alte Spiele ans Fliegen kommen (ADFFS), die beiden Altgeräteemulatoren (ArcEm und ArchieEmu), ein Emulator, der Geld kostet und eher auf RISC OS Select fokussiert ist (VirtualRiscPC), der kostenlose Emulator, der auch RISC OS 5 mag (PRCEmu) sowie das Kleinod, dass unter RISC OS Altsoftware wie Impression Publisher nutzbar macht.

Für mich ist Aemulor hier klar der Sieger – einfach, weil ich das Teil leider immer noch benötige für Impression Publisher und es einfach gut und sauber seinen Job macht.

Setze ich aber mal diese Brille ab, so macht ADFFS, da damit viele der alten Spiele wieder nutzbar sind, was schon eine sehr beindruckende Leistung ist, eine richtig gute Figur, da man speziell bei Spielen gerne bei der Grafikausgabe ein wenig (oder mehr) tricksen musste früher, um auf den alten, langsamen Rechnern ein flüssiges Spielerlebnis zu erzielen.

VirtualRiscPC finde ich zunehmend befremdlich – dabei meine ich weniger, dass der Hersteller damit explizit wirbt, dass es auch unter Windows 2000 und XP etc. läuft (also Wirtssystemen, die man nicht mehr mit gutem Gewissen nutzen möchte), sondern, dass das Teil in der Ära von RISC OS 4.39 stehen geblieben ist – dazu passt auch deren Newsseite, die für 2021 eine Überarbeitung der Webseite und für 2020 den Umzug in neue Büroräume verkündet; die letzte News zu dem Produkt selbst ist aus 2019 (Hinweise zur Mac-Version) und davor geht es einige Zeit nur um die Literatur dort.

RPCEmu hingegen wird aktiv gepflegt, supportet aktuelles RISC OS und kostet obendrein nur die Mühe des Downloads, so dass wenn, dann dieses Kleinod einen Platz auf dem Siegertreppchen verdient hat.

Neuentwicklung

- YouTube-Downloader / Video-Launcher – Chris Gransden (Youtubedl) and Raik Fischer (YTPlay)
- RISC OS Direct – RISC OS Developments mit Ident/Wi-Fi Sheep
- Port der Gennan Deep-Learning/ Artificial-Neural-Network-Library – Paolo Fabio Zaino
- CDVDBurn mit Blu-Ray-Support – hubersn Software
- RISC OS 5.28 mit Support für den Raspberry Pi 4 – RISC OS Open Ltd und alle, die beigetragen haben
- Ein neues Pinboard – RISC OS Developments

Wow, da ist doch mal einiges passiert im letzten Jahr!

OK, was Du hier favorisierst, hängt stark von Deinen Interessen ab. Wer YouTube gerne besucht, dürfte den ersten Kandidaten honorieren wollen; die, die mit optischen Laufwerken nicht nur lesen profitieren sicher von dem neuen CDVDBurn massiv.

Die Audience von Gennan unter RISC OS dürfte, auch wenn's spannend klingt, eher klein sein. RISC OS Direct wäre an sich auch ein interessanter Kandidat, aber der Umstand, dass RISC OS Developments, die dieses Teil verantworten, es nicht einmal fertigbringt, auf der Webseite zu vermerken, welche RISC OS-Version da drin steckt und nur „For Raspberry Pi" dazu vermerkten (auch schon, als es noch nicht den Pi 4 supportete), ist, gelinde gesagt (genauer: geschrieben), befremdlich.

Das Pinboard hat mich hier ein wenig überrascht – einfach, weil es noch nicht fertig ist, denn irgendwie habe ich Vince mal so verstanden, dass das ein Kriterium ist, zumal ich sonst auch Iris hier erwartet hätte.

Für mich ist in dieser Rubrik (Sorry Steffen und Raik) der Umstand, dass RISC OS 5 für den Raspberry Pi 4 fit gemacht wurde, das Highlight.

Initiative

- Livestreamen der Southwest Show – Wi-Fi Sheep mit den Showorganisatoren
- RISC OS Build Service – Gerph
- RPCEmu „easy start bundles" – Matthew und Peter Howkins
- Wohltätigkeitsaktion, um das Ausfallen der Show mit Charitystand abzufedern – WROCC
- Hosten einer Online-Show samt parallelem Livestream auf YouTube – ROUGOL
- Arbeit am neuen TCP/IP ohne dabei die entsprechende Bounty abzugreifen – ungenannt, aber dank RISC OS Developments

Wow, auch hier einige echt gute Dinge.

Was die gestreamten Shows betrifft, ist das natürlich fantastisch und ich würde es begrüßen, wenn es das in gewissem Umfang auch nach Corona geben täte, weil wir so kostengünstiger direkt mitbekommen können, was sich so auf den Shows tut. Zu Zeiten der Acorn World Show, die mehrere Tage dauerte und riesig war, lohnte die Reise; aber die aktuellen Shows, die nur ein paar Stunden dauern und eher übersichtlich sind, locken weniger an – aber wenn die zukünftig parallel online besuchbar sind, kann es für Aufwind sorgen.

Die Bundles von RPCEmu finde ich zwar für mich überflüssig, aber grundsätzlich eine hervorragende Idee, weil es Interessierten den Einstieg massiv erleichtert, zumal mit der neuen Netzwerktechnik, die er Emulator bietet, ein virtueller RISC OS Rechner in Windeseile am Start ist.

Der TCP/IP-Stack ist natürlich ein (noch nicht fertiges) Schwergewicht und definitiv eine gute Sache. Was nur leider noch unklar ist, ist, wann der verfügbar wird und ob er in einer Form implementiert wurde, die für RISC OS Open als Bestandteil in RISC OS dann auch akzeptabel ist.

Website oder Onlineressource

- Webseite Grafikprogrammierung in BBC BASIC – Richard Ashbery
- RISC OS Build Service – Gerph
- RISC OS Open Forums – RISC OS Open
- The Icon Bar – The Icon Bar
- Stardot-Foren – Stardot
- Acorn News Service

Huch, da ist doch tatsächlich ein Kandidat erneut nominiert. Anyhow dürfte hier – und das nicht zu Unrecht – das Forum von RISC OS Open gewinnen; warum hier RISCOSitory nicht genannt wurde, dafür aber durchaus passivere Angebote, ist mir nicht so ganz klar (außer vielleicht, dass der Macher der Umfrage sich nicht selbst nominieren wollte).

Publikation oder Offlineressource

- Impression-Newsletter – Chris Hall
- VirtualRiscPC in Use und Supplements – T. O.M. S.
- Drag 'n Drop – Christopher Dewhurst
- Archive Magazine – Gavin Smith
- Acorn - A World in Pixels – Idesine
- The WROCC – WROCC

Offenbar nur UK-Angebote hier – und bitte gestattet mir, dieses mal nicht die einzelnen Angebote zu kommentieren. Ein wenig schade ist, wen man alles nominieren musste, um zumindest ein halbes Dutzend Kandidaten zu haben.

Fremdsprachliche Ressource

- RISC OS Berlin
- Big Ben Club-Website
- GAG-News
- RISCOS.fr
- Steffen Huber's Blog

OK, vier Webseiten „gegen“ eine Zeitschrift, die seit fast 30 Jahren zweimonatlich erscheint – und damit die mit großem Abstand am längsten regelmäßig erscheinende Publikation ist, die ihre Termine hält – sorry, I couldn’t resist.

Show oder Event
Wie in der Einleitung geschrieben, gibt es hier keine Kandidaten – aber ich hätte einen Vorschlag, auch, wenn er irgendwie „pervers“ ist: Corona!

Grund ist, dass wir eben wegen der Pandemie mal bequem und preiswert an Shows oder Treffen in UK teilnehmen können. Und wenn die Organisatoren das Potential erkennen und auch zukünftig zumindest die Shows oder auch auf Clubtreffen interessante Gäste per Onlinestream oder gar Onlineteilhabe weltweit zugänglich machen, das wäre durchaus reizvoll.

Innovatives/interessantes Projekt

- Pläne zur Überarbeitung des Soundsystems – Jason Tribbeck dank Kontakt mit Cloverleaf
- RISC OS Cloverleaf’s Crowdfunding via Kickstarter – RISC OS Cloverleaf
- RISC OS Build Service – Gerph
- Iris Browser – RISC OS Developments
- DARIC – Daryl Dudey

Ah, hier ist er endlich, der Browser Iris – und das sogar in durchaus potenter Gesellschaft, was die Wahl gerade nicht einfacher macht. Leicht überrascht bin ich, dass der Build Service nun schon das dritte Mal nominiert wird – konnte sich Vince nicht entscheiden (oder fand er das Posting davon, dass am 1.4.2020 erschienen ist, so gut)?

Klar, ein guter Browser fehlt schon lange, aber gerade das letzte Jahr hat gezeigt, dass es auch eine andere „Baustelle“ gibt, wo RISC OS verbesserungspotential bietet und dass auf einem Browser aufsetzen kann: Onlinekonferenzen und somit neben dem Browser auch Support für Headsets und Kameras – im USB-Umfeld.

Cloverleaf hat auch einige nette Projekte in Planung und durchaus Potential, dass wir interessante Soft- und Hardware für RISC OS bekommen – also auch ein durchaus würdiger Kandidat.

Beigetragen habender
Wie üblich nur ein Eingabefeld für den Namen von jemandem, der viel für RISC OS getan hat. Erfahrungsgemäß folgt in einiger Zeit dann eine „Ausfüllhilfe“ – aber wen man hier wählt, hängt auch davon ab, was man mitbekommt, da einige eher dezent im Hintergrund Dinge tun, andere hingegen mit viel Sichtbarkeit in der Öffentlichkeit dabei sind, wobei bei einigen deren Sichtbarkeit größer ist, als der Beitrag – und alle Abstufungen dazwischen gibt es auch.

Broken cog
Hier musste ich dann mal etwas mehr übersetzen – hoffentlich hat das neutral geklappt:

- Das immer häufiger auftretende Problem, dass Ankündigungen an ein eingeschränktes Publikum gesendet werden, was dazu beiträgt, dass das Betriebssystem und seine Entwickler- und Benutzergemeinschaft weniger lebendig aussehen.

- Für das Missverständnis von Lizenz- und Urheberrechtsanforderungen und Aussperren von Andrew Poole aus seiner Facebook-Gruppe, um das dort anzusprechen (obwohl sie später das Richtige getan haben) – RISC OS Cloverleaf.

- Der Status vieler Websites von RISC OS-Unternehmen, die, wenn sie nicht so aussehen, als gehörten sie in die 90er Jahre, häufig veraltet sind und nicht aktualisiert werden.

- Software, die auf problematische Weise bereitgestellt wird, was potentiell zu Konflikten mit anderer Software und gemeinsam genutzten Bibliotheken führen kann.

- Das Fehlen einer anständigen, modernen, integrierten Entwicklungsumgebung.

Der „offizielle“ Kanal für Ankündigungen ist wohl immer noch im Usenet comp.sys.acorn.announce und damit eine alte Technik, die zunehmend schwieriger zu nutzen ist, weil Server, die Usenet anbieten, nach und nach weniger werden. RISC OS Open hingegen nutzt das eigene Forum und mir scheint, Vince will sich so darüber beklagen, dass man ihn nicht aktiv informiert. Leider ist RISCOSitory nicht gerade auf dem Niveau von drobe und veröffentlicht auch gerne mal nicht (wie ich unlängst lernen durfte).

Die Kritik an Cloverleaf ist schon grenzwertig und warum hier auf jemandem so herumgehackt werden muss, der zwar mal einen Fehler gemacht und dann korrigiert hat, aber an sich RISC OS wirklich helfen will, dürfte die freuen, die dieses Mal nicht an dieser Stelle „geehrt“ werden sollten.

Die Kritik an den Webseiten vieler Firmen ist sicher berechtigt, wobei die für die RISC OS Awards da bestens mit reinpasst und ebenfalls Cog-kompatibel ist.

Der indirekte Seitenhieb auf Iris, dass eine Shared Library dabei hat, die in der Betaphase nur für Iris taugte, kann man machen, aber warum.

Dass wir keine Entwicklungsumgebung haben, die den heutzutage üblichen Komfort bietet, ist bekannt und sicher nicht hilfreich…

Fazit

Wie üblich kann man im zweiten Teil noch diverse Dinge zu sich selbst, seiner RISC OS-Hardwarelandschaft, dem Budget etc. angeben und vielleicht wertet Vince diese Zahlen auch dieses Mal nach einigen Jahren des Ignorierens der Rückmeldungen aus.

Bin gespannt, da dieses Mal zumindest in einigen Kategorien kein so klarer Sieger zu erwarten ist. ❍

Drei Dutzend

Diese News hatte Anfang April mit 28 Seiten an sich einen guten Umfang und natürlich hie und da ein wenig Luft, um das einfließen lassen zu können, was sich bis zum Druck an interessanten Dingen ergibt und auch der Option, ggf. einen kürzeren, zeitlosen Beitrag zurückzustellen oder vier Seiten nachzulegen.

Doch dann kamen nicht nur die RISC OS Awards „um die Ecke“, sondern die Wakefield Show hat mich inhaltlich deutlich positiv überrascht – daher haltet Ihr (immerhin zu einer Art Jubiläum mit Ausgabe 175) 36 Seiten in Händen und ich habe (zumindest gefühlt) die Freiheit, die nächsten Ausgaben zum Ausgleich (auch der Kosten) ein wenig dünner zu machen.

Wakefield Show
24.4.2021

Herbert zur Nedden

Dank Corona war die Wakefield Show eine Serie von Vorträgen – und zwar bis auf eine kurze Mittagspause durchgängig von 10 bis 18 Uhr –, wobei ich das im Besuchermodus via YouTube getan habe. Zeitlich passte der Termin fast perfekt zum Erscheinungstermin dieser News – allerdings mit der kleinen Nebenwirkung, dass diese News daher und weil kurz vorher auch die RISC OS Awards online gingen, spontan dicker geworden ist.

Erst bekamen AMGOG Games sowie die drei „großen", sprich RISC OS Developments, R-Comp (Interactive) und RISC OS Open je eine Stunde Zeit im Theatre; dann folgten ein halbes Dutzend halbstündige Präsentationen, wahrscheinlich, damit möglichst viele die Gelegenheit bekamen zu reden.

Hier die Informationen der Anbieter:

AMGOC Games

10:30 www.amcog-games.co.uk:

Spy Mission
Ihr neues Spiel „Spy Mission: The Ice Caves of Dr. Atom" wurde der Öffentlichkeit präsentiert und zu RDSP und dem AMCOG Development Kit gab es ebenso ein paar Worte wie einen Überblick über ihre Spielesammlung.

Anthony Bartram meinte zum neuen Game, dass ein wenig Zufall mitspielt, so dass offenbar zwei Durchläufe nicht gleich sind. Neu ist auch der Joystick-Support, den sie auch nach und nach ihren Altspielen unterjubeln wollen. Die Demo war für mich optisch und akustisch Geld sparend. Bei einem Game nannte er die Endmusik das Highlight :-)

RDSP
RDSP, der virtuelle Soundchip, soll mit dem nächsten Release noch einfacher zu nutzen werden, weil man u. a. auch „lauter" oder „schneller" als Steuerungselement nutzen kann.

Und...
Anthony zeigte dann ein Game nach dem anderen, was erklärte, warum er eine ganze Stunde Zeit brauchte.

RISC OS Developments

11:39 www.riscosdev.com:

Da Andrew Rawnsley zwei Firmen vertritt, bekam er direkt einen Doppelslot. Im Namen der ersten Firma waren Iris, der neue TCP/IP-Stack, Pinboard 2 und ein paar andere Kleinigkeiten Thema.

RISC OS Developemts
RISC OS Developments wurde gestartet, um einen Browser zu liefern und hat dann zwischenzeitlich RISC OS selbst erstanden mit dem Ziel, dass es mehr Verbreitung findet und mit Projekten wie RISC OS Direct scheint sich etwas in der Richtung zu tun.

Pinboard
Ein Ziel, dass mit dem Pinboard 2 adressiert wird, ist RISC OS-Usern etwas Neues zu bieten, ohne bekannt Gutes zu zerstören, aber neuen Usern Dinge zu bieten, die die sie gewohnt sind. Nett ist, dass man Icons auf der Pinwand fixieren kann, sprich versehentliches Verschieben wird verhindert. Gut ist bei Icons auf dem Desktop, dass man die Schrift mit Umrandung zeichnen kann, damit auch weißer Text auf weißem Hintergrund erkennbar ist. Pinboard ist noch Beta – zu Recht, da es während der Präsentation den Rechner abgeschossen hat. Andrew hat die Zeit genutzt, um die Optionen und Features recht detailliert zu präsentieren – schon recht beeindruckend. Wer Pinboard 2 schon mal ausprobieren möchte, möge Andrew eine E-Mail senden und bekommt dann den Download.

TCP/IP
TCP/IP-Stack, basiert auf modernem OpenBSD-Stack mit IP v4 und v6 und für WiFi etc. bietet er die Basis. Da DHCP noch nicht ganz tut, da der Teil Features nutzt, die RISC OS an sich bis dato nicht kannte, ist der Stack noch nicht verfügbar. Aktuell sind hardwareseitig die ARMX6-Treiber fit und die für den Raspberry Pi 4 fast fertig; weitere folgen. Stack ist sehr kompatibel, damit alte Software damit tut – und Teile des neuen Stacks harmonieren auch mit dem alten Stack und das API ist so gleich, wie geht (sprich Software kann auf altem Stack mit nur v4 und auf neuem mit v4 und v6 laufen, ohne dafür angepasst zu werden). Der neue Stack soll bald über die Nightly Betas von RISC OS Open kommen (was natürlich für die R-Comp-Systeme nix bringt, da die ein R-Comp-RISC OS brauchen).

Iris
Iris ist nun auch für OBrowser-Kunden zugänglich und hat ein JavaScript-JIT; OBrowser ist jetzt via PlingStore verfügbar und wer es kauft (und damit RISC OS Developments supportet) bekommt, wenn ich es korrekt verstanden habe, auch Iris in der Betaversion. Iris wird, wenn stabil genug, kostenlos verfügbar. Google Maps samt Streetview funktioniert und auch ein Electron-Emulator in JavaScript in Iris unter RISC OS lief auch. Für Iris gilt, CPU-Performance hilft.

Shared Libraries
Andrew wurde in der Frage & Antwort-Session darauf angesprochen, dass es schon lästig sei, dass Iris mit einer eigenen SharedLibrary daherkommt, die mit der aktuellen insofern inkompatibel ist, dass, wenn Iris einmal gelaufen ist, z. B. Python nicht mehr laufen würde, ohne neu zu booten. Er entschuldigte sich un klärte auf, dass Iris eine neue Shared Library benötigen würde (u.a., weil der JIT für JavaScript mit Thumb-Instruktionen arbeitet, die sich massiv von der aktuellen unterscheide und viele Funktionen erstmals nach RISC OS bringen würde. Es wäre schon aufwändig genug, das mt Iris sauber ans Fliegen zu bekommen aber selbstverständlich würde diese Baustelle vor einer Freigabe von Iris behoben werden. Dabei merkte er auch an, dass aktuell nur recht wenige Applikationen die SharedLibraries nutzen würden (viele Ports verwenden lediglich Unix-Home, dass auch in neuerer, aber kompatibler Version dabei ist, aber nicht diese Libraries), so dass es wohl ein überschaubarer Aufwand ist, wenn die neuen Library sauber tut, das glatt zu ziehen. Auch hätte Raik schon erfolgreich die alte und neue gemergt und auch das scheine zu funktionieren.

RISC OS Direct
Der Plan, auf Pi-Jams RISC OS-Direct-Karten zu verschenken und so neue User zu gewinnen, ist durch Corona verzögert worden; daher haben sie stapelweise SD-Karten herumliegen. Auch konnte RISC OS Developments, wie Andrew ausführte, Interesse aus der ganzen Welt wecken.

Und...
Noch erwähnt hat er Python 3; es ist *„bascially the Basic of 21st century"* und LanMan98/FS, das demnächst SMB v3 bieten soll (womit Unicode mit ins Spiel kommt, was das Filerbackend von RISC OS nicht kennt und daher eine Zeichensatzkonvertierung eingebaut werden muss). Zu Impression-X soll später im Sommer News kommen.

64 Bit
Was die Zukunft von RISC OS betrifft, ist der 64-Bit-Betrieb etwas, was nun am Horizont ist, zumal neue ARM-Prozessoren schon jetzt zum Teil den 32Bit-Modus nur „unter Protest" bieten, quasi. Ob das via Aemulor-artiger Lösung oder mit einer Art Hypervisor zwischen RISC OS und der Hardware oder was – wir dürfen gespannt sein.

R-Comp (Interactive)

12:45 www.rcomp.co.uk:

Mit dem Hut seiner eigenen Firma auf ging es mit einer Viertelstunde Verspätung, da für RISC OS Developments die Stunde einfach nicht gereicht hat, um R-Comps Hard- und Software:

4té
4té, nun auch mit USB-C-Port, ist gut nachgefragt und mit 250 Pfund die günstigste Maschine von R-Comp. VNC ist beim 4té in den Basistools mit drin; damit ist der Rechner auch ohne Monitor („headless") einsetzbar, wobei die Auflösung des nicht-existenten Monitors einstellbar ist (der Pi detektiert bekanntlich normalerweise automatisch, was für ein Monitor angeschlossen ist). 4té-Tools gibt es als PiTools im PlingStore, die u. a. auf dem Pi 400 optional F10 die Rolle von F12 übernehmen lassen, da der bekanntlich F11 und F12 nur per Doppeltastendruck bietet. Angedacht ist für rund 100 £ die 4té Softwaresammlung ohne die Hardware anzubieten (PiTools-Käufer bekommen Upgradepreis) und ein Turbo-4té mit Overclocking etc. ist in Arbeit.

PiTX
Die Plattenperformance des Raspberry ist dank USB 2 träge. PiTX ist ein Board für den Pi, dass auf dem Pi Compute Module aufsetzt und via PCI-Express-Switch diverse Hardware anbindet, wozu auch ein 4-Port-SATA-Controller (NVMI lohne beim Pi nicht) neben den eh von Pi 4 bekannten Ports gehört. Design steht; Problem ist aktuell die Beschaffung der Bauteile und die Herstellung an sich, wobei eventuell nicht nur grundsätzliche Lieferengpässe sondern auch der BRExit Schuld sind.

Und...
Zu sonstiger Software wie dem neuen Fireworkz mit „dynamic data sharing", worüber wir schon berichtet haben, kam er kaum, da RISC OS Developments „überzogen" hat. Nach ein paar Infos zu NAS war der Zeitslot vorbei.

RISC OS Open

14:00 www.riscosopen.org:

Steve Revill ging auf die Entwicklungen rund um RISC OS sowie die Bounties ein.

Historie
RISC OS Build für den Pi 400 mit Python, Updates für DDE und das BBC Basic Handbuch war ausverkauft, wurde daher überarbeitet und als Ausgabe 3 ist es nun wieder verfügbar (auch bei Amazon) und kostenfrei Downloadbar.

Bounties #1
ChangeFSI und Paint können PNG ausgeben, ersteres mit diversen dafür üblichen Optionen. PNG-Ausgabe sei in Applikationen einfach implementierbar, da es als ein Modul vorhanden ist, dass dem schon lange verfügbaren für JPEG vom Handling her ähnlich ist.

Wichtig sei die Bounty zur Toolbox-Vereinheitlichung; seinerzeit gab es ja die Zweige von RISCOS Ltd mit Select und Castle mit 32 Bit. Ein nativer GIT-Client ist in Arbeit und für die Entwicklung von RISC OS sehr hilfreich, da es den Umweg über „Hilfssysteme" erspart. Auch würde so ein Client es anderen erleichtern, ihre Projekte auf Git zu hosten was es, für den Fall, dass deren Arbeit später bei RISC OS Open Eingang finden soll, deutlich erleichtern kann.

Software
MoreDesk ist nun Open Source, weshalb bei ePic CloudFS dessen „Platz" eingenommen hat; Sample Player (Gegenpart zum AMPlayer) für WAV etc. ist bei RISC OS Open gelandet und gehört nun zu RISC OS.

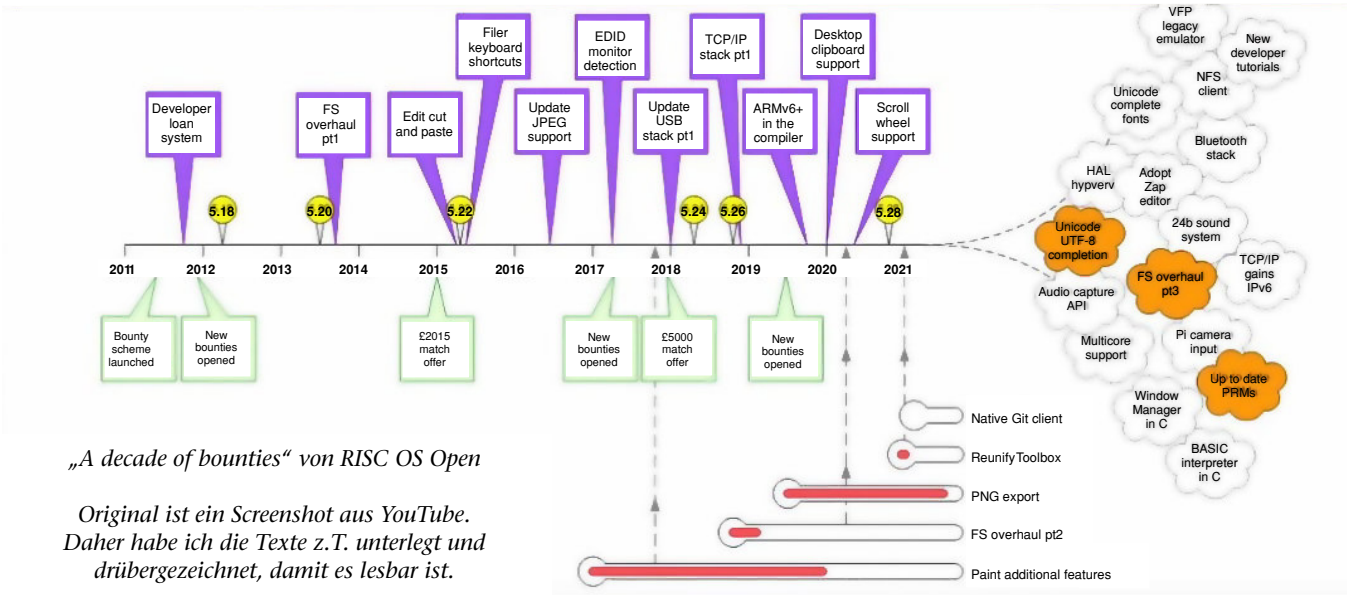
Bounties #2
Die Bounties gibt es sein zehn Jahren und sie haben ca. 81 k£ bewegt. Sieben große sind fertig, fünf in Arbeit, drei hinten runtergefallen und vier warten auf Restfinanzierung. Bei Bounties, die es nicht geschafft haben, geht das Geld an die offenen Bounties. In Bezug auf die Zukunft hat RISC OS Open viele Ideen von denen UTF-8, Filingsystem Teil 3 und ein Update der PRMs im Fokus stehen. Speziell UTF-8 ist nicht ohne. Die nächste Runde für das Filingsystem ist auch eventuell ein neues Format, bei dem aber die RISC OS-Attribute weiterleben. Bei den PRMs ist viel zu tun – im ersten Schritt gilt es in *ein* konsistentes Werk ohne Add-Ons, Errata, … zu bringen.

Zukunft
Was die Zukunft betrifft will RISC OS Open der „Hub" der Community bleiben und die Entwicklung weiter begleiten/steuern; es ist für sie schon toll, dass alle „Guten" mit denen reden und die Koordinierung soll das Spalten, das es mal gab, verhindern.

Für die Zukunft nannte Steve sechs Topics:

1. Mehr RAM (also jenseits der 4 GB).
2. BBC Basic mit Augenmaß erweitern, damit es noch BBC Basic bleibt um u. a. Strukturen und dynamischen Speicher.
3. Das abzusehende Ende der 32Bit-Welt: Einige ARM v8 bieten schon jetzt nur 64 Bit; ARM v9 soll kommen und da wird 32Bit wenn, nur schwer zugänglich sein.
4. Multicore-Support um arbeitslose Kerne zu nutzen; die Verlagerung der Zeropage war ein erster Schritt dafür.
5. Multi-Monitor-Support; ein paar Boards haben die Ports, aber das OS muss es selbst können, damit u. a. Errorboxen nicht halbe-halbe erscheinen.
6. Vector Floating Point, sprich die Nutzung neuer ARM-Kern-Features kann der Rechenperformance helfen; das muss auch der C-Compiler direkter supporten.

USB 3
Zu USB 3: Als die Bounty geschrieben wurde, war es im NetBSD-Stack, aus dem der von RISC OS aufsetzt, noch sehr rudimentär; nun da USB3 bei NetBSD recht gut drin ist, dürfte der Text der USB Stack Bounty Teil 2 *„if deemed sufficiently mature, this could include USB3"* positiv bescheiden sein.



„A decade of bounties“ von RISC OS Open

Original ist ein Screenshot aus YouTube. Daher habe ich die Texte z.T. unterlegt und drübergezeichnet, damit es lesbar ist.

Sine Nomine Software

15:00 sinenomine.co.uk:

VNC
Bei den Demos von Matthew Philipps zeigte sich, dass sich RiscOSM und VNC gelegentlich beharken (VNC kam zum Einsatz, um den RISC OS-Desktop via Zoom auf der Show durchreichen zu können).

RiscOSM
RiscOSM Version 2.0 – Hauptgrund der neuen Versionsnummer ist ein neues Datenformat (wobei das Altdatenformat noch tut, aber einige der Verbesserungen nicht greifen können, so dass es schon Sinn macht, die Daten komplett zu erneuern).

Speziell die Verarbeitung der Karten bei geringer Vergrößerung ist schneller geworden, indem Dinge, die eh nicht dabei visualisiert werden, möglichst gar nicht erst abgerufen werden.

Sie zeigten einen lokalen Miniwebserver unter RISC OS, an den GPS-Tracker mitteilen, wo sie sind und unter RiscOSM konnte man denen live folgen (der Router musste dazu natürlich den Traffic erlauben und richtig durchreichen).

Und...
Recce, das Tool zum Abrufen von Wetterinfo, interagiert auch mit RiscOSM.

In den Restminuten des Slots erwähnte Matthew noch, dass Impact die Datenbankstruktur visualisieren und ImpMail nun u. a. Blindkopien kann.

Elesar

15:30 www.elesar.co.uk:

CloudFS
Erst ging Robert Sprowson auf die Updates von CloudFS (kann nur die pCloud ansprechen), bei dem einfach wählbar ist, ob die Daten in der EU oder US liegen und dass einen Link-Keepalive eingebaut sowie es bis zu dreimal so schnell ist wie früher. Es ist nun in ePic von RISC OS Open enthalten.

WiFi HAT & TextEase
Dann warf er einen kurzen Blick auf Prophet, zeigte WiFi HAT und als Abschluss das Konzept, dass bei TextEase, mit dem er auch die Präsentation für die Show gemacht hat, das „Easy“ im Namen absolut berechtigt ist, da es supereinfach zu nutzen sei.

CJE Micro's

16:00 www.cjemicros.co.uk:

Photodesk
Last (aber hier am Anfang für einen bessere Spaltenumbruch vermerkt) but not least verkündete Chris, dass Paul Reuvers von X-Ample Photodesk übernommen hat, was Entwicklung und Marketing betrifft.

Hardware
Chris Evans startet mit wenig Hardware für historische Rechner; u. a. rollradlose Mäuse und Ersatzkabel für Altmäuse und ging über zu ihren Rechnerchen.

Pi mit SATA
CJE Micro's hat auch Pläne für einen Pi mit SATA, wobei es nicht nur für RISC OS gemacht wird, sondern für eine andere Zielgruppe (was sich auf den Preis für uns positiv auswirken kann – sprich es ist eventuell preiswerter, als die Lösung von R-Comp), aber was da technisch genau geplant ist, war ein wenig unklar.

BRExit
Als Händler leidet er unter dem BRExit, da auch bei Sendungen in die EU ggf. bei Einfuhr MwSt/Zoll fällig werden, was einige Kunden entsetzt; aber auch die Gegenrichtung impliziert VAT-Fälligkeit. Und wenn der Verkäufer wie 4D nicht VAT-Registered ist, wird trotzdem MwSt fällig.

Soft Rock Software

16:30 www.softrock.co.uk:

Escape from Exeria
Vince Hudd zeigte sein Game „Escape from Exeria“. Es war das erste Spiel, dass er geschrieben hat – auf A3000 in BBC Basic. Es ist eine PacMac-Puzzle-Kombination. Spiel war schnell codiert aber das Leveldesign dauerte Wochen.

Die neue Version ist mit hochauflösender und ansprechender Grafik und dezentem Sound. Einige Level sind fies – also gut angelegte fünf Pfund, da man lange brauchen wird (sofern man eine gewisse Hartnäckigkeit mitbringt).

Cloverleaf Project

17:00 riscoscloverleaf.com:

Ziele
Stefan Fröhling stellte die Ziele von Cloverleaf vor: mehr User, Hardware, Treiber, Software, GUI und RPCEmu-Verbesserungen. Der Rockchip RK3399 ist im Fokus, da es mit dem Chip auch einen Laptop gibt; dabei arbeiten sie mit R-Comp zusammen. Bei der GUI hoffe ich inständig, dass er seine Idee, dass ein Klick irgendwo in ein Fenster dieses zwingend nach vorne holt, optional ist – das ist eine der besseren Features vor RISC OS und unter Windows nervt just dieses Verhalten immer wieder.

Projekte
So ist mit CodeCube eine integrierte Entwicklungsumgebung und mit ArtCube eine Bildverarbeitung im Portfolio neben dem bekannten ChatCube, für das Clients für Andriod und iOS sowie IRC und Signal auf dem Zettel sind. Die Systeme basierend auf dem RK3399 sind inkl. 3D-GPU-Support in Arbeit. Und was die GUI-Modernisierung betrifft, will er mit Paolo Zaino zusammenarbeiten, der just dafür ein Projekt am Start hat.

Bei Filer soll das Handling bequemer werden; so soll es möglich sein, beim Fallenlassen von Objekten auf einem Ordner, die Dateien etc. dort hineinzukopieren und die Tastatursteuerung ausgebaut werden, damit nicht immer die Maus notwendig ist. Ferner soll die Thumbnailanzeige integriert werden.

Bei RPCEmu ist u. a. Rollradsupport und eine gemeinesamt Zwischenablage mit dem Wirt (allerdings nur Windows und Linux) im „Angebot".

Kickstarter
Neuer Kickstarter mit insgesamt 20 Rewards ist fast am Start. Eine Frage war, ob es nicht zu viele Ziele seien und man sich nicht dabei leicht verzetteln könne; da aber einige Dinge nicht von Stefans Team gemacht werden, relativiert sich das ein wenig… Auch zeigt sich ja evtl. so, wo das Interesse groß ist, und wo nicht (aber einiges wollen sie auch machen, wenn das Geld nicht reicht – mutig, dass er das so offen sagt…). Und dann meinte er, alle Programmierer müssen mitziehen, sprich einige der neuen Funktionen in ihren Programmen auch nutzbar machen; wow…

Da der Kickstarter noch nicht am Start ist (und ich nicht warten will, damit diese News fertig wird), da Stefan noch ein paar Dinge nachjustieren muss, habe ich mir die einzelnen Rewards, die Cloverleaf auf www.kickstarter.com/projects/riscos-cloverleaf/risc-os-built-the-future-os-for-your-powersaving-computer auslobt, für Euch notiert:

1. Kitten Pi (Rechner, der eher für neue User gedacht ist)
2. Puma RK3399 (Desktop-Gerät in verschiedenen Gehäusen)
3. Laptop RK3399 (PinePro Laptop)
4. RK3399 SD Card (sprich vor allem die Treiber für die, die schon so ein Gerät haben)
5. Cloverleaf RISC OS Distro (ähnlich RISC OS Direct mit ein wenig anderer Software)
6. Cloverleaf RISC OS Distro Pro (ditto, aber mit mehr kommerzielle Software)
7. ArtCube (Bilderverarbeitung)
8. ChatCube mit IRC-Support
9. ChatCube mit Signal-Anbindung
10. ChatCube für Android und iPhone
11. CodeCube: Integrierte Entwicklungsumgebung basierend auf GCC.
12. Desktop-Verbesserungen (im Kern Paolo's Projekt)
13. Filer-Verbesserungen
14. RPCEmu-Verbesserungen
15. NVMe-Treiber
16. WiFi-Treiber
17. Art-Bundle (ArtWorks & ArtCube)
18. 2D/3D GPU-Treiber, initial für den RK3399
19. Reward Nummer 19 hat er nicht vorgestellt – vielleicht gibt es den ja gar nicht
20. Super Sponsor, sprich für die mit viel Geld: gib einfach mal 1k€

Ambitioniert – aber interessant.

Lustig bei der Präsentation war zu sehen, dass er u. a. einen Ordner mit Lesezeichen zu Donald Trump hat.

RISCOSbits

17:30 www.riscosbits.co.uk:

Gehäuse
Andy Marks zeigte diverse Gehäuse und Zubehör, die zum Teil durch Einfügen von Layern in der Höhe variabel sind. Sie sein meist aus Kunststoff, aber es gibt auch einige aus oder mit Holz.

FOURtress
Bei (der) FOURtress setzen sie auf ein Standard-Aluminiumgehäuse mit regelbarem Lüfter, so dass sie den Pi mit mehr als 2GHz betreiben können. Da eine M2SSD als Laufwerk dient wird die SD-Karte nicht nur zum Booten genutzt, sondern auch für automatische Backups.

PiAno
Der neue PiAno ist weniger Performant, dafür aber preislich als „Einstiegsdroge" positioniert. Man kann die Systeme mit wenig oder (gegen Aufpreis) diverser Software bekommen und eine OS-Upgrade-App haben sie auch am Start – einfach gesagt: enduserfreundlich.

Dual-Boot
Fast fertig und schon zu sehen war deren Dual-Boot-System mit RISC OS und Linux, bei dem ein Klick auf einen Pinguin auf dem RISC OS-Desktop den Rechner auf einen Linux-Desktop durchbootete; und das Zahnrad dort drehte den Spieß wieder um.

Mein Eindruck ist, dass mit RISCOSBits nunmehr ein weiterer Anbieter von direkt nutzbaren Rechnern am Start ist.

Fazit

Es war nicht unanstrengend, so lange mit Kopfhörern vor dem Rechner zu lauschen (und das hat nichts damit zu tun, dass es Englisch war, da das für mich kein Problem ist) – aber es hat sich gelohnt und war streckenweise hochinteressant. Hilfreich war dann allerdings, dass man den Stream auch jetzt noch ansehen kann, denn obgleich ich mitgeschrieben habe, war es gut, nochmal nachhören zu können.

Alles in allem: Gelungen! ❍

GAG DVD 175 ☆
Ich habe die, die RISC OS-Distributionen anbieten (wollen), angeschrieben und gefragt, ob sie mir eine aktuelle für die GAG DVD senden wollen. Dabei habe ich auf der Webseite von RISC OS Cloverleaf, RISC OS Developments und RISC OS Open (Reihenfolge ist alphabetisch und somit nicht wertend) nach der offiziellen Kontaktadresse gesucht und die angeschrieben, statt explizit bestimmte Personen der drei.

Cloverleaf hat geantwortet, dass deren Distribution als kommerzielles Produkt erscheinen wird, womit sie nicht kostenlos auf die DVD kommen kann. RISC OS Developments und RISC OS Open haben mir gar nicht geantwortet (ich hatte natürlich in der E-Mail eine Deadline für die Antwort vermerkt und eine Erinnerungs-E-Mail gesandt).

8GB
Jeffrey Lee hat mal wieder ein wenig an RISC OS verbessert, damit RISC OS auf dem neuen Raspberry Pi die 8 GB RAM erkennen und einrichten kann. Nun fehlt nur die Software abgesehen von einer RAM-Disc, die diese Unmengen von Speicher sinnvoll nutzt :-)

TextEase
Die neue Version 5.98 bringt der Logo-Komponente weitere Filingsysteme neben ADFS und CDFS (man glaubt es nicht) bei und kann SpecialFX für augenfälligere dünne Linien nutzen.

VATGST
Mit dem BRExit entfällt auch die Prüfung von UK-Steuernummern auf den Seiten der EU; Kevin Wells (kevsoft.co.uk) hat daher ein Tool geschrieben, dass curl verwendet nun UK-Quellen dafür anzapft, aber auch die der EU, damit man nicht von Hand an zwei Stellen schauen muss.

TBX C++
Die C++ Toolboxlibrary TBX C++ ist bei Version 0.7.6 angelangt und nicht mehr „alpha“, da es nun seit 10 Jahren genutzt wird: sites.google.com/site/alansriscosstuff/tbx. Die neue Version unterstützt eine Reihe weiterer Toolboxfunktionen und hat eine Reihe zusätzilcher Features erhalten.

SERVstat und VECstat
Die beiden Monitoring-Tools für Service- und Vectorcalls wurden um ein paar neue derartige Calls, die es in RISC OS 5 gibt, erweitert. Saugbar von armclub.org.uk/free.

DARIC ☆
Steffen Huber

Mitte März habe ich – Pandemie sei Dank – im Rahmen des monatlichen Online-Meetings der RISC OS User Group of London einem interessanten Vortrag von Daryl Dudey lauschen können. Er stellte sein Projekt DARIC vor, das kurz gesagt eine BASIC-artige Programmiersprache für Windows und RISC OS nebst Laufzeitumgebung basierend auf einer virtuellen Maschine darstellt. Mit einem starken Fokus auf die Grafikfähigkeiten, im Geiste von Klassikern wie AMOS (und natürlich dessen Cousin STOS) oder BlitzBasic, aber jetzt auch mit 3D-Grafikunterstützung. Syntaktisch mit deutlicher Verwandtschaft zu BBC BASIC.

Da ich derartige Vorträge zur Entspannung anhöre und nicht zum Zweck der Blogberichterstattung, habe ich natürlich keinerlei Notizen gemacht. Alles, was jetzt folgt, basiert auf der fehlerhaften Erinnerung eines mittlerweile in die Jahre gekommenen Gedächtnisses. You have been warned.

Die Kernpunkte von Daryls Bemühungen, quasi die dahinterliegende Philosophie, ist für mich gar nicht so interessant. Er strebt eine sehr interaktive Sprache an, also ein klassischer Interpreter mit einer integrierten Ausführungsumgebung mit Zeileneditor, wo man die Befehle direkt eintippen oder auch ausführen kann. Außerdem soll die Sprache reichlich Schlüsselwörter bieten, mit allem was das Programmiererherz begehrt („*batteries included*“, wie Daryl es nannte), ohne dass man erst zig Bibliotheken runterladen muss. Also durchaus BBC BASIC nicht unähnlich, aber insbesondere für Einsteiger noch nutzerfreundlicher.

Klare Zielrichtung: 2D- und 3D-Grafikunterstützung – derzeit alles in Software gerendert – und mit identischen Ergebnissen egal, ob die Ausführung unter RISC OS oder Windows erfolgt.

Also: da bin ich nicht Zielgruppe. Den Kreativbereich im Grafikbusiness sollen andere abdecken, ich bin eher so der Anwendersoftware-mit-grafischer-Oberfläche-Entwickler. Für mich sind andere Details interessant. Zum Beispiel die Tatsache, dass Daryl „mal kurz“ eine eigene VM entwickelt hat, die zudem recht performant zu sein scheint. Und in einer Hochsprache (C++) implementiert. Mit Ansätzen eines JITs. Mit einem Parser, der per ANTLR4 generiert wird. Letztlich wird hier der Nachweis erbracht, dass mit modernen Werkzeugen ein solches Projekt – Sprache, VM, Laufzeitumgebung – selbst für einen einzelnen Entwickler im Bereich des Machbaren liegt. Wobei natürlich zu beachten ist, dass es vom weitgehend funktionierenden Zwischenstand hin zu einer wirklich ausgereiften stabilen Version noch ein ganzes Wegstück sein wird. Die Grundidee „schnelle VM mit erweitertem BASIC“ ist jedenfalls für RISC OS-Zwecke hochinteressant.

Sehr erfrischend fand ich Daryls Pragmatismus an vielen Ecken. Er hat sich nicht jahrelang Gedanken über sprachtheoretische Feinheiten oder eine effiziente GC-Implementierung gemacht, oder ob man nicht besser eine vorhandene VM portieren sollte oder, oder, oder. Er hat einfach mal angefangen zu implementieren, Irrwege in Kauf genommen, mit einer klaren Philosophie im Hintergrund, und was dabei rausgekommen ist, kann sich sehen lassen. Auch interessant: einfach auf C++ 14 als Implementierungssprache gesetzt, was unter RISC OS schon anspruchsvoll ist, weil es GCC 8 voraussetzt.

Wer den Stand der Dinge begutachten mag: der Sourcecode ist auf GitHub verfügbar: github.com/zolbatar/Daric. Da kann man auch diverse kleine Beispiele und Demos in Augenschein nehmen – nichts illustriert Syntax besser als lebender Code. Die Website zur Sprache unter www. dariclang.com scheint leider gelegentlich offline.

Witzige Randnotiz zum Abschluss: Der Erfinder von AMOS ist gerade auch am Schrauben und hat AOZ Studio (www.aoz.studio) aus der Taufe gehoben, ursprünglich „AMOS 2“ genannt. Ich traue mich noch nicht, die große BASIC-Renaissance auszurufen, aber es könnte sich hier ein Trend abzeichnen.

Backuplaufwerk
Ein User nutzt eine große SSD für seine Windows-Backups und wollte die auch für RISC OS verwenden. Eine Idee ist, sie mit FAT32 zu formatieren, was dann allerdings nur dann Sinn macht, wenn unter Windows keine zu großen Dateien mitspielen. Alternativ schlug Steffen vor, die Platte zu partitionieren: Vorne eine FAT32 für RISC OS und hinten eine NTFS für Windows. Man kann nach meinem Dafürhalten auch mit LanManFS von RISC OS aus auf die Platte sichern und den Windows PC als Vermittler verwenden.

LanMan98 ☆
LanMan98 Version 2.08 ist Open Source – es liegt auf www.riscosdev. com/lanman98 und im Plingstore – unter der Common Development and Distribution Licence 1.0. Diese Lizenz impliziert, dass Änderungen abzuliefern sind und ebenfalls dieser Lizenz unterliegen (also eine Regelung wie wir sie für RISC OS 5 haben, nur ohne Karenzfrist).

LanMan98 ist ein Ersatz für LanManFS, dass schon länger in RISC OS enthalten ist, und war früher sein Geld wert, weil es vor allem schneller und dann auch funktional besser als LanManFS war, aber zwischenzeitlich sind die Unterschiede geringer.

Es folgt in der Bekanntmachung noch:

„We would like to dedicate this release to members of the RISC OS community who are no longer with us, especially in the light of a tough 12 months. In particular, we remember past members (and thank remaining members) of the RISC OS Investment Group (RIG) who have supported ongoing networking work on RISC OS, making projects like this possible.“

Damit wird das Release all den Mitgliedern der RISC OS-Community gewidmet, die verstorben sind – vor allem in Hinblick auf die harten letzten zwölf Monate. Besonders gedacht wird den verstorbenen Mitgliedern der RISC OS Investment Group (und den verbliebenen wird gedankt), die die Arbeiten am Netzwerk für RISC OS unterstützten und Projekte wie dieses ermöglichten.

Bitte beachte, dass die Formulierung *„who are no longer with us“* als auch die *„past members“* beide definitiv Verstorbene und nicht etwa solche, die sich abgewendet oder umorientiert haben, meinen. Im Englischen wird das so formuliert (auch, wenn Google Translate & Co. das nicht so übersetzen).

Iris
Unlängst postete ein User auf der ARMini-Mailingliste, dass er begeistert sei, weil eine neuere Ausgabe von Iris nun Dinge im Zusammenhang mit dem Onlinebanking gut kann.

Was ich lustig fand ist, dass Andrew auf eben dieser Liste regierte: *„Also, for the record, people shouldn't really be talking about new versions of Iris - they are alpha/beta, and not public knowledge - the material in our newsletters is supposed to be confidential.“* Köstlich :-)

Iris ☆
Die Entwicklung von Iris geht weiter und RISC OS Developments sind bei Version 1.010 lt. Newsletter angelangt.

Die lokal abgelegte (und somit anpassbare) Startseite um ein paar Suchmaschinen erweitert – unlängst war da eine Diskussion auf einer Mailingliste, dass einige Google nicht sehen wollten.

Auch noch nicht so lange dabei scheinen ein paar Einstelloptionen wie die Startseite, Zugriff auf die Historie und Bookmarks zu sein. Eine Kennwortverwaltung ist nicht in Sicht, was bedauerlich ist, da ohne eine solche für mich ein Browser unvollständig ist; dabei sei angemerkt, dass ich bei fast jedem Onlineshop, wo ich etwas kaufe, einen Account anlege mit individuellem, perversem Kennwort der Art „_Zfa$xpyUh?x3pYYqxu8“, was unter Windows mit KeePass als Primärablage sowie für das Gros das Sichern der Zugangsdaten im Browser (nicht für kritische Dinge) total bequem tut.

Neu ist, wenn man den Onlineinformationen Glauben schenken darf, der Support von WebFonts, Local Storage und FileUpload, so dass Iris nun auch für eBay und PayPal fit ist.

Mit 1.010 wurde der Just-in-Time-Compiler für JavaScript aktiv, was dem Browser auf die Sprünge hilft, und die Speicherverwaltung in der UnixLib wurde neben diversen anderen Dingen in Iris selbst verbessert.

TLS ☆
TLS 1.0 und TLS 1.1 sind nun offiziell „depreciated“, wie auf im RFC 8996 (datatracker.ietf.org/doc/rfc8996) verkündet. Das macht sie zu einem echten No-Go für den Einsatz, zumal damit jedwede Sicherheitsfixes etc. vom Tisch sein, weil „gibt keine mehr“.

TLS 1.2 wurde 2008 zum Standard erklärt und TLS 1.3 folgte 2018.

Wenn somit R-Comp ihren Messenger Pro nicht mit dem aktuellen TLS für RISC OS ausliefert, sondern dem Uralt-TLS, bietet es formal kein TLS mehr :-)

Prophet
Version 4.08 bietet ein paar Ausfüllautomatiken und hebt Rechnungen vor, wenn die VAT fehlt. In Bezug auf „Making Tax Digital“ folgt der Dialog mit HMRC (also den offiziellen Seiten) der aktuellen Spezifikation v3.0.

IP-Stack ☆
Kaum war News 174 gedruckt, schrieb Andrew Rawnsley, dass RISC OS Developments bekanntlich einen neuen TCP/IP-Stack entwickelt habe, um RISC OS eine moderne, pflegbare Netzwerkplattform zu bieten, die auch IPv6 und WiFi unterstützen würde – und in Bälde ein paar „Beta Spots“ für i.MX6-User verfügbar werden, um den neuen Stack auf Her(t)z und Nieren zu testen (O-Ton: *„to give this a thrashing“*).

Die, die sich dazu bereiterklären, sind gebeten, den Stack wirklich zu nutzen und testen und ein wenig Know-How in Sachen IP wäre hilfreich. Aktuell noch limitiert auf i.MX6; erst wenn die Basis stabil sei, sollen weitere Plattformen dazukommen – einfach, um Doppeltarbeiten beim Einspielen von Korrekturen zu vermeiden.

WiFi ist noch nicht drin – Ziel sei, erst einmal den alten Stack von Acorn zu ersetzten, wobei IPv6 schon tun würde.

Ich bin gespannt… ob der Stack von RISC OS Developments dann auch in die Konzepte und Anforderungen von RISC OS Open passt und somit allen zugutekommen wird. Und auch, ob ich beim „Thrashing“ mitmachen darf.

DDE ☆
RISC OS Open hat DDE v30a herausgebracht – wie üblich kostenfrei für die, die in der letzten Zeit für DDE etwas ausgegeben haben; als gebührenpflichtiges Upgrade für die anderen.

Der Assembler ObjAsm wurde massiv erweitert und unterstützt alle ARM-Instruktionen bis hin zum ARM v8.6, womit auch Fließkommaoperationen mit reduzierter Genauigkeit Einzug halten, die bei Technologien wie AI (Artificial Intelligence) zum Einsatz kommen. Der neue Datentyp, BFloat16 ist gewissermaßen ein auf Float16 gekapptes Float32, weil für den Exponenten derselbe Platz da ist und daher die Mantisse kürzer, sprich der Wertebereich ist der „große“, die Genauigkeit wurde beschnitten, damit die Zahl weniger Bit belegt, was der Performance sicher zugutekommt.

Aber auch an anderen Stellen hat RISC OS Open einiges für DDE v30 gegenüber den Vorgängern verbessert, womit der Basic-Compiler funktional deutlich zugelegt hat und der C-Compiler den ISO-Standard ISO9899:2018 erfüllt, der eigentlich C17 heißt, aber offenbar zunehmend auch C18 genannt wird.

WebDAV
Unter Windows kann ich den Onlinespeicher bei meinem Provider bequem als Laufwerk mappen, wobei der Pfad zum Speicher z. B. das Format https://xxx.1und1.de hat.

LanManFS und LanMan98 können zwar Shares im Internet mappen, aber weder verschlüsselte noch WebDAV oder via http, sondern nur das alte CIFS, sprich NetBEUI oder nacktes IP.

Cavern
Ein Spiele-Klassiker ist, wie ich lesen durfte, Bubble Bobble – ein Plattform-Game, dass in einer Höhle gespielt wird, wo Du durch die Gegend hüpfst und Dinge abschießt. Für dieses Spiel ist für den Pi eine Python-Implementation im Code the Classics-Buch unter dem Namen Cavern enthalten. Wir können uns die Mühe des Abtippens sparen, da Jeroen Vermeulen das Game unter Verwendung des AMGOG Development Kits implementiert hat.

Die Screenshots auf RISCOSitory haben ein klein wenig irritiert, da sie relativ (!) gut aussehen. Zu finden ist Cavern im Plingstore.

YTPlay ☆
Raik Fischer

YTPlay liegt aktuell hier: www.riscos.berlin/download/YTPlay_V310RC1.zip. Ich empfehle, das aktuelle ffmpeg von Chris Gransden, weil das gut 10x schneller als die älteren Versionen tut: www.riscosports.co.uk/eabi

Wenn die Suche im Suchfenster (Select aufs Symbolleistensymbol) erfolgreich war, kann man wie gehabt per Klick:

- ➤ Select spielt den Listeneintrag ab.
- ➤ Spezial macht 'nen Download.
- ➤ Umschalt-Spezial übergibt die URL an ffmpeg. Das erzeugt ein mp3 mit Artist- und Titel-Tag sowie (fest hinterlegter) Coverart.

Via Spezial-Klick auf das Symbolleistensymbol kann man youtubedl und youtubesearch installieren und updaten. Je nachdem von wo und wann man Python installiert hat, braucht es manchmal zwei Anläufe, bis es klappt.

Python ☆
Python 3.8 gibt es schon länger für RISC OS (via PackMan); nunmehr ist eine Beta von Python 3.10 von Chris Johns unter packages.lessthan3.org.uk/pkg/Python-310-3.10.0-1.zip saugbar.

USBScopePlus ☆
!USBScopePlus ist !USBScope ähnlich, kann aber nicht nur einlesen (vlugo: capture) und analysieren, sondern auch ausgeben: www.audiomisc.co.uk/software. Auf seinem ARMX6 klappt es mit 192k/24 Bit Stereo Ein- und Ausgabe gleichzeitig; welche andere Hardware das leistet, weiß er nicht.

Wenn ich mich daran erinnere, dass ich mit einer 8MHz Z80-CPU zwei parallele serielle Übertragungen mit 19.200 Baud im Hintergrund absolut stabil laufen lassen und den Rechner derweil für Textverarbeitung benutzen konnte, finde ich Dinge wie das oben geschriebene immer ein wenig lustig.

!USBScopePlus kann wahlweise mit einem bidirektionalen Gerät arbeiten oder mit zwei verschiedenen. Bei zweien ist wichtig, dass sie (zumindest derzeit) in der richtigen Reihenfolge am USB-Bus hängen, da die Applikation ungefragt das erste, passende nimmt. Neben USB Audio Class 2-DACs (Digital-Analog-Wandler) harmoniert das Tool nun auch mit Class 1.

Neu ist auch, dass man seine eigene „Waveform" hinterlegen kann und eine namens „Wave From Hell", die einen DAC so richtig quälen kann, ist nicht nur mit dabei, sondern auch mit Vorsicht zu genießen. Für die, die sich einlesen wollen hat Jim auf www.audiomisc.co.uk/HFN/OverTheTop/OTT.html einiges geschrieben.

!USBScopePlus benötigt als Unterbau die USB Audio Library von Colin Granville (ftpc.iconbar.com/usb).

Backplane
Einige erinnern sich noch: Die Backplane ist die Platine, die man in Rechner von Acorn steckte und die wiederum Steckplätze für Podules bietet – und Podules aka Erweiterungskarten gibt es ja bekanntlich einige. Beim Risc PC waren je nach Bauhöhe Backplanes für zwei oder vier Podules gängig, die zum Ein- oder Zweislicer gehören; aber es gab da ja auch die Option, noch ein paar Slices mehr zu spendieren und daher gab es sogar längere Backplanes.

CJE Micro's bietet nun ebensolche für den A7000 und A7000+ an, die Platz für ein Podule bieten. Sie Backplanes bieten zusätzlich einen I²C-Anschluss sowie optional eine Echtzeituhr. Für 49 Pfund gibt es die uhrenlose; mit Uhr kostet es ca. 20 Pfund mehr.

QrCode
Kevin Wells hat QrCode Version 1.08 herausgegeben (kevsoft.co.uk). QrCode ist ein Tool, mit dem man die bekannten und vor allem im Umfeld von Smartphones beliebten QR-Codes zu erzeugen. Die neue Version kann nicht mehr als die alte, bietet aber ein paar Verbesserungen im Userinterface und benimmt sich anständiger. Auch erkennt es, wenn wget nicht vorhanden ist, und weist darauf hin. Wget ist ein Kommandozeilentool zum Abrufen von Webseiten; QrCode nutzt es, da es sich des Webdienstes auf goqr.me bedient.

SWI400A1
Anton Reiser hat sich mit dem SWI Font_ScanString befasst und das zur Veranschaulichung in die Applikation !SWI400A1 gepackt: home.allgaeu.org/areiser/riscos/software.

Der SWI simuliert gedanklich die Ausgabe eines Strings mit Font_Paint und liefert Informationen über den String. So ist er hilfreich, um die Position des Caret im String zu finden, oder wo er umgebrochen werden sollte.

Bekomme keinen Schreck, wenn Du die Applikation startest – dieser SWI hat es parametertechnisch in sich – was nicht überrascht, da der SWI Font_Paint ebenso flexibel ist, was sich in dessen Parametern zeigt. Die Rückgabe wirkt übersichtlich, da sie einen Zeiger in den String sowie den Koordinatenoffset auf dem Bildschirm dazu liefert – mal fürs Caret und mal für die Umbruchstelle.

MoreDesk
RISC OS supportet bekanntlich große Monitore (wenn auch einige Hardware da gewisse Grenzen hat); der Betrieb zweier Monitore ist funktional noch ein wenig rudimentär und nur auf wenig Geräten möglich. Reicht der Platz nicht, greift man zu dem, was früher fast normal war: Einem Tool, mit dem mehrere Desktops hat, zwischen denen man flugs wechseln kann.

Von 7thsoftware.co.uk/moredesk.html kannst Du MoreDesk kostenlos saugen. Es kommt mit einem Installer, da es Hilfsmittel wie Routines und ConfiX benötigt. An sich recht nett ist die Idee, dass in den Quellen für die Installation die !Boot-Dateien der Applikationen ~Boot genannt sind, damit nicht aktiv und beim Installieren umbenannt werden, sobald die Applikation da ist, wo sie hin soll.

Cat
Bei dem Namen !Cat denkt man heutzutage wohl tendenziell an Katzen, die speziell mit diversen Bildern und Filmen in den sozialen Medien sehr gefragt sind. Doch der geneigte RISC OS-User weiß, dass *Cat das ist, was auf anderen Plattformen „dir“ oder „ls“ ist, sprich das Kommando zum Anlisten eines Verzeichnisses; in ihrer unendlichen Weisheit hat Acorn einige gängige Kommandos und Zeichen anders belegt, da wir den Punkt als Verzeichnistrenner und *Dir als Kommando zum Wechseln des aktuellen Verzeichnisses haben, was im Mainstream der „cd“ (change directory) leistet.

!Cat, das in Version 0.29 auf www.svrsig.org/software/Cat.htm zu finden ist, zeigt den Inhalt eines Verzeichnisses grafisch in Baumansicht an.

Nett ist das Feature, dass !Cat sich für ein Verzeichnis einen Schnappschuss merken kann und damit dann später aufzeigen, was sich dort geändert hat. Auch kann man nach Dateinamen filtern, sollte dem Tool aber kein Archiv als „Verzeichnis“ spendieren, das es das nicht so wirklich mag.

Awards ☆
Auf riscosopen.org/forum/forums/5/topics/16251 wurden die Awards diskutiert. Da sind so komische Dinge zu sehen, dass ADFFS rausfliegen soll, da es nichts für „backward compatibility“ tut – so ganz kann ich der Argumentation nicht folgen, dient es doch, alte Spiele spielbar zu machen. Python 3.8 soll, weil noch zu unfertig, nicht als Entwicklerwerkzeug nominiert werden.

Beim Broken Cog war ein Item, dass sich Vince beschwert, dass andere wir RISC OS Open ihre neue Software nur auf ihrer Webseite bekanntgeben, statt ihm für sein RISCOSitory zu huldigen und ihn aktiv zu informieren :-)

Anyhow, ADFFS ist drin geblieben und obiges „Broken Cog“ wurde ohne Nennung von RISC OS Open formuliert. Die Abstimmung für 2020 nun online und soll ein paar Monate laufen.

DeskWatcher ☆
DeskWatcher, die kommerzielle Screensharinglösung von Thomas Milius, ist bei Version 0.08 angelangt. Ein paar Bug sind weg und DeskWatcher kennt „inverse“ Verbindungen, was eventuelle Anpassungen in Firewalls oder auf Routern unnötig machen kann, und bietet einen Filetransfer.

GIT
Die RISC OS-Quellen wurden vom alten CVS auf das neuere GIT umgestellt. Dummerweise ist GIT nicht gerade simpel und GIT-Software für RISC OS aktuell eher dünn gesät – daher gibt es eine Bounty just dafür.

Grund überhaupt ein Repository zu nutzen ist, dass es eine Versionskontrolle und Transparenz bei der Verwaltung des Quellcodes bietet – speziell, wenn mehrere Entwickler an dem Code arbeiten, was bei RISC OS der Fall ist, ist so ein Tool sehr hilfreich, zumal es auch erkennt und aufdeckt, wenn zwei Programmierer denselben Code verändert haben und in dem Fall hilft, beides übereinanderzuziehen.

Die GIT-Bounty ist mit ca. 4k£ dotiert – und das Geld ist da, so dass sie nun im Status „underway“, also in Arbeit ist.

Discobolus
Steve Drain hat das Modul Discobolus herausgegeben, dass die Plattenzugriffe schreibender und lesender Nuancierung in Zeigernähe visualisiert. Anlass war, dass viele der neuen Laufwerke wie SD-Karten etc. kein „Blinkelämpchen“ mehr haben. Und bei einigen Operationen erscheint gar die Sanduhr – gerne mal farbig. Auf kappa.me.uk findest Du Version 0.07 des Kleinods.

Tipp: Probiere es aus, bevor Du es beim Booten automatisch lädst – bei mir mit RPCEmu unter RISC OS 5 hatte ich nach dem Laden des Moduls Hänger.

DeskEdit
DeskEdit liegt nicht im Quellcode vor, so dass die für aktuelle Hardware fitte Version durch Patchen entstanden ist. Unter aktuellem RISC OS zeigt sich, dass in ein paar optischen Ungereimtheiten der (funktionierenden) Savebox.

Moan
Schon ein lustiger Name für ein Spiel, denn „to moan“ bedeutet nicht weniger als „zu stöhnen“. Es ist ein Desktop-Kartenspiel für bis zu vier Spieler, bei dem es gilt mehrere Level nacheinander zu schaffen – so kann es die Aufgabe sein, zwei Drillinge zu sammeln und sich seiner Restkarten zu entledigen… wenn ich die leicht unklare Info auf 7thsoftware.co.uk/moan/index.html korrekt verstanden habe.

WiFi HAT
Version 1.06 von EtherWILC behebt ein paar Bugs und aktualisiert ggf. die Firmware des Netzwerk-Coprozessors.

OSLib
Wer mit der OSLib arbeitet und genau liest, hat es einfacher, da beim Linken mit DDE die Library OSLibSupport vor der OSLib angegeben werden muss. Bei C-Quellcode kenne ich es, dass der Compiler nur Dinge versteht, die er kennt; aber dass beim Linken die Reihenfolge der Angabe der Libraries relevant ist, war mir zumindest neu.

Avalanche
Auf github.com/effarig/ro_avalanche/releases ist die Beta 2.05 des VNC-Clients Avalance zu finden. Nur ein kleines, eher heimliches Update mit dem Ziel, die Stabilität zu erhöhen.

Maestro
Einige erinnern vielleicht noch Maestro, dieses alte Tool zum Zugehörbringen von Dingen. Im Repository von RISC OS Open stolperte ich doch tatsächlich über diesen Exoten, da diese Applikation in den Genuss einiger Korrekturen kommt. Ich ahnte nicht, dass das Teil noch eine Rolle spielt.

Statistisch
Dass 5 Millionen Geimpften üblicherweise 15 aber nun leider 30 an Nebenwirkungen Verstorbenen bedeutet, dass die Mortalität um 50% gestiegen – kann man so schreiben und die Presse tut das auch, da die Information, dass 0,0006% statt 0,0003% betroffen waren, ja nix hermacht. Ob man seriös andeuten kann, dass die Corona-Mortalität eher bei 1% liegt (RKI-Zahlen), um die 0,0006% zu relativieren, überlasse ich den Fachleuten und denke mir meinen Teil… Trau keiner Statistik, die Du nicht selbst gefläscht hast, sagt man – genau hinschauen, wer sie beauftragt hat und was somit Zweck ist, kann den Blick oft schärfen.

Kurz notiert zu Kurz Notiert
Die Reihenfolge der Beiträge in der Rubrik „Kurz Notiert“ hat keinerlei Signifikanz. Vielmehr versuche ich sie so anzuordnen, dass die Spalten gut gefüllt sind und die einzelnen Einträge immer in einer Spalte leben, wozu ich die Textblöcke munter hin und her schiebe, bis es passt. Dazu kommt, dass ich meist kurzfristig, wenn das Gros der News schon fertig gelayoutet ist, noch brandaktuelles einfüge, sofern interessant genug, und dabei ggf. zeitloses zurückstelle oder hie und da Texte ein wenig straffe. Dazu kommt, dass das, was ich für wichtig halte, eventuell für Dich eher nebensächlich sein kann. So ist’s neutraler…

Laufzeitvariablen unter RISC OS

Das Modul ExtdVars

Herbert zur Nedden

Von Thomas Milius gibt es ein putziges Modul namens ExtdVars (voller Name „Extended Variables“), dass ein paar erweiterte Systemvariablen bereitstellt, die keinen festen Wert haben, sondern im Moment der Abfrage oder per Kommando betankt werden. Mit dem Modul kannst Du zwei unterschiedliche Variablenarten definieren, die dann in der klassischen Form als solche unter RISC OS nutzbar sind, sofern das Modul ExtdVars geladen ist.

Das eine ist eine Kommandovariable, die mit dem Sternchenkommando ExtdVars_CreateCommandVar angelegt wird. In diesem Fall soll die Variable ein Kommando enthalten, denn beim Auswerten der Variable wird eben dieses Kommando ausgeführt und die erste Zeile der Ausgabe des Kommandos wird als Variablenwert geliefert.

Für die, die aus der Computerurzeit das Spiel „Eliza“ kennen, kann die zweite Variable hilfreich sein, die mit dem Befehl ExtdVars_CreateQuestionVar erzeugt wird. Hierbei wird eine Frage (mit ein paar Optionen) hinterlegt, die Frage per Wimp_ReportError an den geneigten Anwender gestellt und dann dessen per Klick gegebene Antwort in die Variable geschrieben.

Dritter im Bunde der netten Kommandos ist ExtdVars_Repeat, mit dem die Zeilen einer Textdatei kommandomäßig durchgeflöht werden können. Hierbei gibt man als Parameter neben dem Namen der Variablen die erste und letzte Zeile und natürlich noch den Pfad zur Textdatei an. Die Zeilennummern starten bei Null und Zeile -1 ist die letzte der Datei.

Für die, die es eher einfacher haben wollten, sind zwei Variablen schon eingebaut:

ExtdVars$UUID liefert, wie der Name klar und deutlich suggeriert, einen „Universally Unique IDentifier“ (UUID) gemäß RPC 4122 (tools.ietf.org/html/rfc4122). Diese kommen z. B. beim Logging von Events aber auch als in iCalender-Dateien als eindeutige Kennung zum Einsatz.

Da ist ExtdVars$ActualDir schon eingängiger, da es den Pfad zum aktuellen Verzeichnis (currently selected directory) liefert, sprich das, was per *dir gesetzt wird. Diese Variable ist z. B. bei Kommandozeilen-Prompts nutzbar, damit man direkt sieht, was Sache ist. Ja, liebe Mainstreamsystenmutzer, das *dir von RISC OS entspricht dem cd-Kommando auf anderen Plattformen und das, was Windowsuser als dir kennen, gibt es unter RISC OS als *cat.

Nur aus Jux probiere doch mal dieses Kommando als Einzeiler (mit Leerzeichen getrennt) ein:

```
setmacro
   CLI$Prompt
   <ExtdVars$ActualDir>
   (<Sys$Time>)
   *
```

Schon ist der Kommandoprompt im CLI aufgewertet, da er erst das aktuelle Verzeichnis (sprich das Current Directory, dass du mit *Dir setzen kannst) und in Klammern die Uhrzeit zeigt.

Dieses Kommando

```
ExtdVars_Repeat x 0 -1
   <Boot$Dir>.!Run %Echo <x>
```

simuliert den Befehl

```
Type <Boot$Dir>.!Run
```

… aber statt der einfachen Ausgabe der Zeile per Echo kannst Du natürlich auch alles andere mit den Zeilen anstellen und so Textdateien wahlweise als Quelle für Kommandoparameter verwenden (um z. B. eine Liste von Dateien zu verarbeiten) oder auch durchsuchen etc.

Per (wieder Einzeiler)

```
ExtdVars_Question
   Geschlecht
   2304
   "Eine Fräge hätte ich"
   Was bist Du?|Männlich,Weiblich,Divers
```

fragst Du das offensichtliche und bekommst in der Variablen Geschlecht (erster Parameter) das Ergebnis – und zwar als Buttonnummer und nicht in der Textform des Buttons. Die Zahl, die als zweiter Parameter im Kommando genannt ist, ist das Flag, dass der SWI Wimp_ReportError erwartet, hier allerdings in dezimaler Syntax, der Dritte ist der Titel und der Rest die Frage nebst den Buttons.

Alles in allem ein nettes Kleinod, mit dem man vieles anstellen kann. ❍

Kleinanzeige

Detlef Fröhlich (detlef.froehlich@gmail.com) hat 2x8 & 2x16MB für den Risc PC,l eine 2er Backplane für den Risc PC sowie ein IDE-Interface von MCS für den A310 abzugeben. Alles wurde funktonstüchtig „eingemottet“.

Impressum

© 2021, Herbert zur Nedden Preis für diese Ausgabe € 4,20, Abonnementspreis € 5,35 frei Haus in Deutschland, € 6,70 in Europa.

Herausgeber: Herbert zur Nedden, Alte Landstraße 21, 22962 Siek, Deutschland, Tel. +49 (41 07) 99 00, HzN.2021@HQ.gag.de, www.gag.de, Bankverbindung: Herbert zur Nedden, IBAN DE21 5001 0517 5409 0689 20, BIC INGDDEFF (Bank: ING).

Vertrieb in der Schweiz: Niklaus Weiss, Sagistrasse 20, 6300 Zug, info@weissniklaus.ch; in allen anderen Ländern: Herbert zur Nedden.

Diese Zeitschrift wurde auf einem mini.m unter RISC OS 5 mit Impression/Aemulor erstellt; für Grafiken kamen ArtWorks 2, PhotoDesk, Paint und Snapper zum Einsatz. Für den Druck wurde die Zeitschrift als Postscript 3 ausgegeben, mit GhostScript-Tools in ein PDF in passender Seitenaufteilung gewandelt und auf einen Kyocera P4040dn A3-Drucker geschickt.