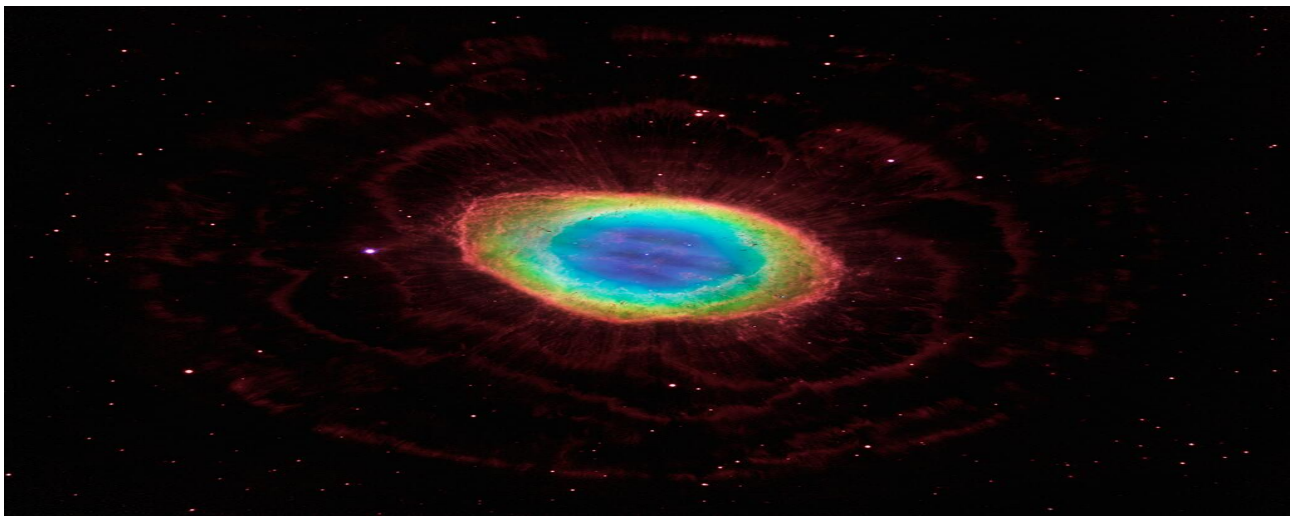

Sternenzeit 20.180.428 - mentales Logbuch

Neues vom Raumschiff mental auf seiner Reise durch Zeit und Raum. Offensichtlich sind wir in eine kosmische Anomalie geraten und haben die normale vierdimensionale Raumzeit verlassen. Und offensichtlich hat uns diese Anomalie nach einem Monat der Irrfahrt wieder ausgespien und wir müssen nun erstaunt feststellen: Es sind in dieser kurzen Zeit fünf Jahr im normalen Universum vergangen. Wie ich darauf komme? Nun, mit Erstaunen habe ich festgestellt, dass der letzte sinnvolle Beitrag im Hive Blog zum Thema **mental** vom **20. Mai 2013** stammt. Wir haben jetzt Mai 2018. Und ich schwöre, ich habe den letzten Blogeintrag gerade erst vorigen Monat (in meiner relativen Zeit) geschrieben!

Und ehrlich, ich habe in diesem Monat ohne Unterbrechung im Maschinenraum des Sternenschiffs **mental** geschraubt und Gizmos ausgetauscht. Ehrlich! Aber der Reihe nach...

Die Idee **mental** mit einem direkten Blockinterface für den Massenspeicherzugriff auszustatten, hatte am Anfang der Reise einige Vorteile und auch einen gewaltigen Charme: Die Routinen für den Massenspeicherzugriff waren verdammt klein und schnell, was dem forthigen Lebensstil der Einfachheit sehr entgegenkommt. Sie standen quasi unter der Überschrift: Zen und die Kunst auf einen Massenspeicher zuzugreifen. :)

Aber leider hat jede Münze zwei Seiten, wenn man mal von der legendären Möbius-Münze absieht. Denn es zeigten sich relativ schnell auch zwei Nachteile:

Auf der einen Seite war die Installation umständlich, da die Tape-Dateien immer unfragmentiert auf der Karte gespeichert sein mußten. Wollte man neben dem **mental** auf der Karte auch noch TriOS beheimaten, waren umfangreiche Kopieraktionen nötig. Wollte man später noch weitere Tapes zu **mental** hinzufügen, musste die Karte quasi komplett neu bespielt werden. Aber das war noch zu akzeptieren, da es nur ein einmaliger bzw. recht seltener Aufwand war.

Das nächste Problem zeigte sich bei der praktischen Anwendung. Auf dem Hive habe ich entgegen meiner alten Gewohnheiten angefangen, als erstes immer irgendwelche

Soundsachen als Demo zu programmieren, früher war das immer ein zellulares System wie das Conway Spiel. Nun, wahrscheinlich liegt das an den guten Soundbiblotheken für den Hive - es ist doch nichts befriedigender, als bei einem frisch programmierten OS erstmal einen DMP Player einzubauen, und in dem Meer an Retromusik zu baden! Da ist ein Conway Game ähnlich unspektakulär wie ein Apfelmännchen... ;)

Leider stellte sich das als nicht so einfach heraus, da ich dafür alle DMPs in das spezielle Containerformat der Tapes umwandeln musste. Unpraktisch und umständlich, liegen die DMP-Dateien ja schon vom TriOS her im FAT16/32 Format auf der Karte. Das nährte meine Zweifel an der Einfachheit der bestehenden Lösung. Denn je länger ich darüber nachdachte, umso größer wurde die Problem: Sollte ich alle DMPs in einen Container packen? Dann müßte ich erstmal ein Tool dafür für TriOS oder auf dem PC programmieren. Was wenn ich später noch einen Datensatz hinzufügen wollte und der Container zu klein ist? Alle DMPs als unfragmentierte Dateien speichern? Irgendwie alles unpraktisch. Ich wollte ja später gern noch mehr mit **mental** machen, da würden sich wahrscheinlich ständig irgendwelche Daten und Dateien ändern. Nun müsste ich ja dann alle Daten in ein solches unfragmentiertes Dateiformat bringen, und spätere Änderungen wären dann auch nicht mehr wirklich einfach.

Für die praktische Anwendung war das unfragmentierte Containerformat also nur scheinbar einfach, denn es hatte sich die Komplexität nur verschoben: Das System war einfach, aber die Nutzung war letztlich komplexer geworden. Eine schlechte Lösung für meine Anwendungsfälle. (Notiz an mich selbst: Die Summe der Komplexität ist immer gleich.) Also wäre es doch besser, den komplexen Teil einmalig im System zu verbergen, um nachher in der Fülle der Anwendungen von der Einfachheit zu profitieren.

Das führte zu einer Radikalkur: Rein in den Maschinenraum und die primitiven Impulstriebwerke gegen ganz neue chromglänzende Wunderwerke ausgetauscht: FSRW raus, FatEngine der neusten Generation "V2.0 - Special" von Kwabena W. Agyeman rein!

Hmm FatEngine, da war doch irgendwas, was ich immer schonmal probieren wollte? Genau: Multifilesupport und Pfadparser! Mehrere Dateien über einen Filedeskriptor parallel öffnen und mit echten Dateipfaden handhaben, um nicht mit den einfachen Filemarkern wie unter TriOS und IOS arbeiten zu müssen.

Gesagt, getan. **mental** kann nun mit maximal 8 Dateien gleichzeitig umgehen. Mit dem Wort **files** kann man sich die Filedeskriptoren und damit die aktuell geöffneten Dateien anschauen. Möchte man zum Beispiel beobachten, wie das System mit den Dateien hantiert, kann man mit dem Wort **watch** die Files ständig überwachen. Das Wörtchen **watch** ruft dabei ein anderes Wort einmal in der Sekunde auf und löscht vorher den Screen, der Wortname muss dabei als String übergeben werden. So kann mit **files watch** das Wort **files** im Sekundentakt aufgerufen werden, um quasi in Echtzeit eine Dateiliste der geöffneten Dateien mit entsprechenden Attributen anzuzeigen. Escape beendet **watch**. Startet man vorher den DMP-Player, so kann beobachtet werden, wie die Musikdateien wechseln und welcher Titel abgespielt wird.

Und mit dem DMP-Player sind wir bei einem weiteren Thema. Unter TriOS kann nicht mehr auf die SD Karte zugegriffen werden, wenn ein DMP-Player läuft. Das konnte am Anfang nur der HSS-Player, da er des Soundmodul komplett in einen Puffer in Administra zwischenspeichert und von dort abspielt. Die DMP-Dateien sind als SID-Register-Stream

mit mehreren hundert Kilobyte aber viel zu groß für eine Zischenpufferung in einem Propellerchip, und können so nur im direkten Dateizugriff abgespielt werden. Der Grund ist: TriOS kann momentan nur eine Datei öffnen.

mental kann nun problemlos DMP-Dateien abspielen und gleichzeitig kann weiter auf andere Dateien zugegriffen werden. Versucht es:

```
dmp                 - DMP Bibliothek compilieren
\sid\1startrk.dmp play - DMP Datei abspielen (Pfad anpassen!)
files watch         - Dateideskriptoren anzeigen
```

oder ein anderes Beispiel

```
dmp                 - DMP Bibliothek compilieren
\sid\1startrk.dmp play - DMP Datei abspielen (Pfad anpassen!)
dir                 - Verzeichnis anzeigen
sys.m use i0        - sys.m öffnen und Index anzeigen
sys.m use br0       - sys.m öffnen und Screenbrowser öffnen
```

Und hier sehen wir schon die zweite Erweiterung im Einsatz: Dateipfade. In der DMP Bibliothek findet sich auch schon eine Anwendung. Dort sind die Pfade der Ordner mit DMP-Datein auf meinen Karten inklusive dem Kommando playdir in ein Wort compiliert:

```
sid1 /sid/sid1 playdir ;
```

Somit entspricht ein solches Wort einer Playliste. Der DMP-Player, oder besser der DMP-Player Wortbaukasten, enthält ebenfalls einige neue und ziemlich coole Bausteine um DMPs abzuspielen:

```
play (cstr--)     - spielt eine einzelne DMP-Datei ab
playdir (cstr--)  - spielt alle DMPs in einem Verzeichnis ab
play+             - nächster Titel im Verzeichnis
pause             - Player pausieren/fortsetzen
stop              - Player stoppen
stereo            - Player im laufenden Betreib(!) auf
mono                stereo/mono schalten schalten
status            - zeigt fortlaufend den Status des Players
```

Zur einfachen Handhabung sind übrigens in der Autostartdatei sys.m alle wichtigen Bibliotheken als Worte angelegt. Als Beispiel compiliert das Wort **dmp** die DMP-Bibliothek also ohne viel Tipperei ins System. Ich habe mir dabei angewöhnt, am Ende einer Bibliothek ein zum Bibliotheksnamen gleichlautendes Wort zu definieren, welches dann die Bibliothek wieder aus dem Wörterbuch entfernt. Das schaut dann so aus:

```
dmp     - compiliert die DMP-Erweiterung
sid1    - Spielt Playliste 1
stop    - Player stoppen
dmp     - löscht die DMP Wörter wieder aus dem Wörterbuch
```

So kann man sehr schnell arbeiten und experimentieren. Aber Achtung: Das zweite

löschende Wort **dmp** löscht natürlich nicht nur die DMP-Wörter, sondern alle Wöter nach der DMP-Erweiterung, da es das Wort **forget** verwendet!

Was habe ich sonst noch so in dem gefühlten Monat programmiert? Im Schnelldurchlauf:

- Der Eingabeparser akzeptiert nun den Präfix % für Binärzahlen und $ für hexadezimale Zahlen.
- Mit der FatEngine gibt es natürlich auch wieder **dir** und **cd** als Wort. Schön forthig: **cd** ist natürlich UPN, also der String steht vor dem Kommando! Also nicht **cd /sid/sid1** sondern **/sid/sid1 cd**. :) Einfach mal die Welt auf den Kopf stellen. Geil, ist wie ein Fahrrad, welches nach links fährt, wenn man nach rechts lenkt - absolut diebstahlsicher!
- In der usr-Datei befindet sich bei mir standartmäßig auf Screen 0 die Definition der Wörter **wld** und **wed** - Abkürzungen für "work-load" und "work-edit". Das Wort **wld** compiliert den Inhalt der Datei usr.m ab Screen 1 (nicht ab Screen 0, dort stehen wld und wed) ins Wörterbuch und entfernt vorher ein evtl. schon vorher compilierten [work]-Abschnitt im Wörterbuch. Ruft man also zweimal hintereinander wld auf, so befinden sich die Worte ab Screen 1 in usr dennoch nur einmal im Wörterbuch. Der praktische Effekt? **wld** compiliert usr.m ab Screen 1 immer "frisch" ins System für schnelle Experimente. Das passende Wort "**wed**" ruft den Editor auf Screen 1 in usr.m auf. So kann man sehr schnell arbeiten. Ist der Code fertig, kann man ihn in eine eigene Bibliothek übertragen. Der Wechsel zwischen editieren und compilieren findet dabei in wenigen Augenblicken mit diesen kurzen Kommandos statt, was "gut von der Hand geht".

Ok, für diese drei Punkte hätte es irgendwie keinen Schnelldurchgang gebraucht, aber was solls. Denn damit sind wir wieder in der normalen Raumzeit angekommen und könne die Reise fortsetzen. Also dann:

Lebt lange und in Frieden.