RWTH AACHEN UNIVERSITY
Betriebssysteme und Systemsoftware
Sommersemester 2014
COM SYS

Übungsblatt 1

Abgabe: 28. April 2014

Aufgabe 1.1: Linux-, Shell und C-Playground (0 Punkte!)

Wir fangen mit ein paar 'Übungen' zum warm werden an, um uns ein bisschen mit Linux vertraut zu machen.

Achtung! Zu dieser Aufgabe gibt es nichts abzugeben und es werden somit auch keine Punkte vergeben.

a) **Get and install Linux**

Zum Bearbeiten der Aufgaben benötigt ihr ein Linux – also einfach das aktuelle Windows-System plattmachen und eine Linux-Distribution runterladen und installieren. ;)
Um sich einen Überblick über verschiedene Distributionen zu verschaffen, kann man hier nachschauen: `http://distrowatch.com/dwres.php?resource=major`

Wenn man nicht weiß, welche Distribution man wählen sollte, sei Ubuntu empfohlen (oder dessen Klon Kubuntu oder Xubuntu, da die Benutzeroberfläche im neuesten Ubuntu auf leistungsschwächeren Geräten seeehr träge läuft).

Wer lieber bei seinem aktuellen System bleiben und/oder nicht gleich einen Dualboot anlegen will, kann eine vollständige Linux-Distribution auch einfach in einer virtuellen Maschine z.B. mittels VirtualBox installieren: `http://www.virtualbox.org/`. Es gibt bereits fertige VirtualBox-Images auf `http://virtualboxes.org/images/`, so entfällt die Installation in der VM.

Installiert auch die sogenannten GuestAdditions mit – sie erlauben z.B. die Einrichtung gemeinsamer Ordner zwischen der virtuellen Maschine und dem Gastsystem in VirtualBox.

b) **RTFM - Read The 'Friendly' Manual!!!**

Wer Probleme mit den Shell- und teilweise auch den C-Aufgaben hat, kann natürlich Google oder Ähnliches zur Hilfe heranziehen – es geht aber auch professioneller. Öffnet dazu eine Shell und gebt die folgenden Befehle ein (ohne $):

```
$ man man
$ man grep
...
```

Hinweis: Read them. Really!

Aufgabe 1.2: Grundlagen der Betriebssysteme (3+1+1+1+1 = 7 Punkte)

a) Beschreiben Sie die Abfolge von Ereignissen, die auftreten, wenn eine ISR durch einen Hardware-Interrupt ausgelöst wird.

b) Bei der Von-Neumann-Architektur spricht man vom sogenannten Von-Neumann-Flaschenhals, um auszudrücken, dass das Verbindungssystem (Daten- und Befehlsbus) zum Engpass zwischen dem Prozessor und dem Speicher wird. Wieso ist dieser Von-Neumann-Flaschenhals in heutigen Rechnersystemen nur noch abgeschwächt von Bedeutung?

c) Welche Vorteile hat es, wenn I/O-Geräte Direct Memory Access (DMA) nutzen?

d) Wann ist es sinnvoll, Polling einzusetzen? Wann macht der Einsatz von Interrupts mehr Sinn?

e) Wie ist es einem Rechner mit nur einer CPU (und einem Prozessorkern) möglich, Prozesse scheinbar gleichzeitig auszuführen?

Aufgabe 1.3: C-Datentypen (1+1+1+1+1+1 = 6 Punkte)

Gegeben seien folgende Deklarationen:

```
char zeile[] = "E  war einmal in einem Land vor unserer Zeit"; // 44 Buchstaben
char* z;
int i = 0;
int* pi;
```

Welche der folgenden Operationen sind möglich? Falls eine Operation nicht möglich ist: warum nicht? Falls sie möglich ist: was bewirkt sie? Gehen Sie davon aus, dass eine Operation das Ergebnis der darauf folgenden Operationen beeinflusst, wenn sie möglich ist.

a) `zeile[1] = 's';`

b) `zeile++;`

c) `pi = &i;`

d) `*pi = 42;`

e) `&i = pi;`

f) `z = &zeile[43]-i;`

Aufgabe 1.4: C-Programmierung (7 Punkte)

Die Datei `obfuscated.txt` wurde mit Hilfe des ROT13-Algorithmus (`http://de.wikipedia.org/wiki/ROT13`) "verschlüsselt". Der ROT13-Algorithmus verschiebt die Buchstaben des Alphabets um 13 Zeichen: aus einem A wird also N und aus einem U wird ein H (Nach einem Z wird bei einem A weiter gezählt, engl.: wrap around). Das Verfahren beachtet nur die Buchstaben A-Z und (unabhängig davon) a-z, keine Umlaute oder Sonderzeichen. Dadurch, dass wir so je genau 26 Buchstaben haben gilt: ROT13(ROT13(x)) = x.

Schreiben Sie ein C-Programm, das den Dateipfad zu `obfuscated.txt` als Befehlszeilenparameter erhält und anschließend auf dessen Inhalt das ROT13-Verfahren anwendet und das Ergebnis auf dem Bildschirm ausgibt. Bachten Sie dabei die folgenden Vorgaben:

- Nutzen Sie zur Verarbeitung der Datei die Librarycalls `open`, `read` und `close`.
- Da bei der Benutzung Fehler auftreten können (z.B. falscher Dateiname), nutzen Sie die Funktion `strerror` zusammen mit der globalen Variable `errno`, um passende Fehlermeldungen auf dem Bildschirm auszugeben.
- Benutzen Sie die Funktion `exit`, falls das Programm durch einen Fehler nicht wie gewünscht weiter ausgeführt werden kann.
- Nutzen Sie die Funktion `printf`, um den Text auf dem Bildschirm auszugeben.

Um die passenden Header-Dateien und die Benutzung der genannten Funktionen zu erfahren, nutzen Sie die sogenannten man pages: `man 2 open`, `man 2 read`, `man 2 close`, `man 3 strerror`, `man 3 errno`, `man 3 exit`, `man 3 printf` und `man 1 printf`. Falls Ihr System nicht alle man pages enthält, ist Google die nächste Anlaufstelle, um die man page zu finden (oder man installiert sie nach).

Das Programm sollte mit `gcc -Wall INPUT.c -o OUTFILE` ohne Fehler und Warnungen kompilieren. Nutzen Sie das als Anhang zum Übungsblatt mitgelieferte Codegerüst, um Ihre Lösung zu erstellen. Sie finden auch die Datei `obfuscated.txt` als Anhang – diese soll lediglich zum Testen der eigenen Implementierung dienen (Kommt etwas sinnvolles raus?); die entschlüsselte Version braucht nicht mit abgegeben zu werden.