Übungsblatt 2

Abgabe bis spätestens 14.11.2016 in Stud.IP

Aufgabe 1 (4 Punkte)

Unter `/home/ti2/ueb/02/aufgabe1` findet ihr einige Dateien mit einem 128 Byte langen ID3v1-Tag[1]. Schreibt ein Shellscript mit dem Namen `mp3-rename.sh`, das (im Rahmen der Möglichkeiten von `bash`) eine beliebige Anzahl an Dateinamen als Argument entgegennimmt und jede dieser Dateien nach folgenden Regeln umbenennt:

- Sind Künstler, Album, Titel und Tracknummer verfügbar, so wird die Datei entsprechend des Musters *Kuenstler-Album-Titel-NN.mp3* benannt. Alle Leerzeichen und Minuszeichen in *Kuenstler*, *Album* und *Titel* werden durch Unterstriche ersetzt. Der Tracknummer *NN* wird eine Null vorangestellt, wenn sie kleiner ist als 10.
- Fehlen entweder Kuenstler, Album oder Titel, so wird das betreffende Feld durch einen Unterstrich ersetzt.
- Fehlt die Tracknummer, so wird der Anteil im Dateinamen *-NN* weggelassen.

Hinweis: Für diese Aufgabe werden Hilfsprogramme wie `tail`, `tr`, `cut`, `echo` und bash-Funktionen wie `read`, `basename`, `while` benötigt. Der „Advanced Bash-Scripting Guide“[2] bietet einen hervorragenden Einstieg in die Programmierung mit bash.

Aufgabe 2 (5 Punkte)

Unter `/home/ti2/ueb/02/aufgabe2` findet ihr ein einfaches Programm zur Berechnung der Fibonacci-Folge. Die Dateien `main.s`, `fib.s` und `show.s` enthalten eine stark vereinfachte Version des Assemblercodes, den der C++-Compiler auf einem LP64-System erzeugt hat. Die verwendeten Symbole für den Linker orientieren sich an der Notation aus der Vorlesung. Der Suffix `m` steht hier für `unsigned long`.

a. Gebt zu jeder der drei Assemblerdateien die Text/Data-Relocation-Table, Symboltabelle und Stringtabelle für das `a.out`-Format an wie in der Vorlesung gezeigt. Geht dazu von der Annahme aus, dass jede Instruktion vier Bytes benötigt, Adressen sind acht Bytes lang. Einträge in der Text/Data-Relocation-Table und der Symboltabelle sind vier Bytes lang.

b. Nehmt an, ein Linker würde die drei Objektdateien in der Reihenfolge `main.o`, `fib.o` und `show.o` als a.out-Datei zusammenbinden. Welche Symbole könnten mit pc-relativen Sprüngen ausgedrückt werden? Gebt die berechneten Offsets für diese Sprünge an!

c. Warum wäre die so erzeugte a.out-Datei so noch nicht ausführbar? Welche Informationen müsste der Linker neben den in Teilaufgabe b berechneten pc-relativen Sprüngen noch hinzufügen?

[1] Siehe https://en.wikipedia.org/wiki/ID3#Layout für eine Beschreibung des Formats.
[2] Siehe http://www.tldp.org/LDP/abs/html/.

Aufgabe 3 (1 Punkt)

Unter `/home/ti2/ueb/02/aufgabe3` findet ihr ein Programm `geheim`, das euch nach einem Passwort fragt. Findet mit Hilfe des Programms *gdb* heraus, wie das Passwort lautet. Dokumentiert eure Vorgehensweise. Warum „weiß“ *gdb* soviel über das Programm, obwohl nur der Maschinencode vorliegt?

Weitere Aufgaben

1. Was machst Du, wenn Dir die genaue Semantik eines Unix-Kommandos entfallen ist?
2. `bla` sei ein ausführbares Programm. Was ist der Unterschied zwischen dem Aufruf `bla` und dem Aufruf `bla &` in der Shell? Welche Auswirkungen hat dies, wenn `bla` von Standard Input liest bzw. auf Standard Output schreibt? Gegeben sei der in Abbildung 1 dargestellte Prozessbaum. Wie ändert er sich nach Eingeben der folgenden Kommandofolge im Shell?

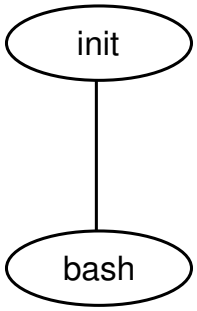

Abbildung 1: Prozessbaum

```
sleep 1000 &
emacs &
bash
date
```

3. Nenne drei Beispiele für Informationen, die der Betriebssystemkern über einen Prozess wissen muss.
4. Was ist eine *Pipe*?
5. Wie macht man ein soeben editiertes Shell-File ausführbar?
6. In welche Bereiche (Segmente) ist der (virtuelle) Adressraum eines Programms in Ausführung in Unix unterteilt, und welche Eigenschaften kennzeichnen sie?
7. Wozu wird der Stack verwendet?
8. Welchem Zweck dienen *Bibliotheken* (*Libraries*)?
9. Welche Aufgabe erfüllt ein *Linker*?
10. Wozu wird beim Assemblieren eine Symboltabelle angelegt?
11. Welchen Vorteil hat es, Bibliotheken mit *Position Independent Code* zu versehen?

Diese Aufgaben müssen nicht abgegeben werden. Sie dienen als Vorbereitung auf das Fachgespräch und werden im Tutorium besprochen.

Abgabe

Bis 24:00 Uhr am 14.11.2016 digital in Stud.IP und – wenn nicht anders mit Eurem Tutor vereinbart – ausgedruckt in unser Postfach in der MZH-Ebene 6 oder im nächsten Tutorium. Es gelten die vereinbarten Scheinbedingungen (siehe Stud.IP). Bitte beachtet unsere ergänzenden Hinweise ebenda.

Packt die abgaberelevanten Dateien in eine Archivdatei. Die Dateinamen innerhalb des Archivs wie auch der Archivname selbst müssen den in den Scheinbedingungen festgelegten Namenskonventionen folgen. Abgaben, die diese Konventionen nicht einhalten, gelten als nicht erfolgt.

Eure Ansätze und der gewählte Lösungsweg müssen nachvollziehbar sein. Achtet insofern auf eine saubere Dokumentation im Quelltext. Benennt alle von euch verwendeten Quellen, auch Zusammenarbeit mit anderen Gruppen und verwendete Unterlagen aus früheren Jahrgängen.

Für Programmieraufgaben ist die Korrektheit der Lösung bzw. deren Grenzen grundsätzlich nachzuweisen. Dies geschieht neben der Dokumentation des Programmcodes durch geeignete Tests, deren Auswahl und Eignung begründet werden muss.