

Übung zur Vorlesung "Grundlagen Betriebssysteme und Systemsoftware"

(Prof. Dr. J. Schlichter, WS 2008 / 2009)

Übungsleitung: Dr. Georg Groh (grohg@in.tum.de)

Tutoren: Dipl. Inform. Vivian Prinz (prinzv@in.tum.de), Dr. Nils Kammenhuber (kammenhuber@net.in.tum.de), Dipl. Inform. Robert Schmohl (schmohl@in.tum.de), Dipl. Inform. Dipl. Geogr. Jan Herrmann (hermanj@in.tum.de), Dipl. Inform. Robert Eigner, David Brodski (brodski@in.tum.de), Yang Guo (yang.guo@gmx.de), Jan Finis (finis@in.tum.de), Martin Levihn (levihn@in.tum.de)

<http://www11.in.tum.de/Veranstaltungen/GrundlagenBetriebssystemeundSystemsoftware0809>

<http://www11.in.tum.de/Veranstaltungen/GrundlagenBetriebssystemeundSystemsoftware0809/uebung>

Blatt 2

- Abgabe: bis 3.11.2008 12:00 Uhr per E-Mail an den Tutor der eigenen Gruppe. Die Mail soll einen Zip-Ordner als attachment haben, der für jede Hausaufgabe einen Unterordner enthält, in dem die Lösung als .txt-File(s), als .c-File(s) o.ä. enthalten ist.
- Beachten Sie bitte das Format fuer den Betreff der E-Mail! (GruppenNr, Name, BlattNr, Tutor, ggf. Zusammenarbeits-Gruppenmitglieder) (Bsp: Gr 1, Müller, Blatt 3, Tutor: Prinz, mit Schmidt und Meier zusammengearbeitet)
- Musterlösungen Hausaufgaben: ab 3.11.2008 12:00 Uhr auf der Übungswebseite zum Download.
- Musterlösungen Tutoraufgaben: ab 10.11.2008 12:00 Uhr auf der Übungswebseite zum Download.

Stoff

Es wird empfohlen folgende Literatur durchzuarbeiten:

- Skript Kapitel 1 (Übersicht) und Kapitel 2 (Einführung)
- Tanenbaum "Modern Operating Systems" (2001): Kapitel 1 (Introduction)

1 Hausaufgabe (Linker in Unix/Linux)

Lernziele

Es soll der Umgang mit dem Linker in gcc am Beispiel exemplarisch ausprobiert werden. Grundsätzliche Einblicke in die Arbeitsweise des Linkers sollen gewonnen werden. Für Studenten, die in die Materie tiefer einsteigen wollen, kann die Aufgabe als Ausgangspunkt für weitere Untersuchungen dienen.

Aufgabe

Es sollen an einer Reihe vorgegebener Aufrufe von gcc die beteiligten Entitäten untersucht und interpretiert werden.

Folgendes Makefile sei vorgegeben:

```
all:
    export LD_LIBRARY_PATH=. #diese Zeile unter Solaris ggf. weglassen
    gcc -S main.c -o main.s
    gcc '-DWORLD="world"' -S helloworld.c -o helloworld.s
    gcc '-DWORLD="universe"' -S helloworld.c -o hellouniverse.s
    gcc -c main.s -o main.o
    gcc -c helloworld.s -o helloworld.o
    gcc -c hellouniverse.s -o hellouniverse.o
    gcc -shared -o libhelloworld.so helloworld.o
    gcc -shared -o libhellouniverse.so hellouniverse.o
    ar cru libhelloworld.a helloworld.o
    ar cru libhellouniverse.a hellouniverse.o
    # Aufgabe 3.1
    gcc main.o helloworld.o -o main1a
    gcc main.o libhelloworld.a -o main1b
    # Aufgabe 3.2
    gcc main.o -L. -lhelloworld -lhellouniverse -o main3a
    gcc main.o -L. -lhellouniverse -lhelloworld -o main3b
    # Aufgabe 3.3
    gcc main.o helloworld.o -o main4a
    gcc main.o -L. -lhelloworld -o main4b
    # Aufgabe 3.4
    gcc main.o helloworld.o -o main5a
    gcc main.o -static -L. -lhelloworld -o main5b
```

weiterhin seien die beiden folgenden C-Programme gegeben:

main.c:

```
#include <stdio.h>
```

```
extern void say_hello();
```

```
int main()
{
    say_hello();
}
```

und helloworld.c

```
#include <stdio.h>

void say_hello()
{
    printf("Hello %s\n", WORLD);
}
```

Alle drei Files stehen zum Download auf der Übungswebseite bereit.

Abgabe

(Siehe Teilaufgaben)

1.1 Teilaufgabe

Kommentieren sie die Zeilen des Makefiles kurz. (Was macht die Zeile? Was ist der Output? Wozu sind die Optionen gut?). Abgabe: Kommentierte Version des Makefiles.

1.2 Teilaufgabe

In einem Satz: Was ist der Unterschied zwischen `main1a` und `main1b`?
Abgabe: Der eine Satz :-).

1.3 Teilaufgabe

Bzgl. `main3a` und `main3b`: Welche der beiden Libraries wird jeweils aufgerufen? In einem Satz: Warum gibt es keinen Konflikt zwischen den Namen?
Abgabe: Zwei Sätze.

1.4 Teilaufgabe

In einem Satz: Wie unterschieden sich `main4a` und `main4b`? Benutzen Sie zur Beantwortung zusätzlich das Linux/Unix-Tool `nm`. In maximal 3 Sätzen: Warum besitzt das Symbol `say_hello` beim einen Mal eine Speicheradresse und beim anderen Mal nicht?
Abgabe: Die beiden Outputs von `nm` für die beiden Versionen und die (maximal) 4 Sätze.

1.5 Teilaufgabe

In maximal 2 Sätzen: Wie unterschieden sich `main5a` und `main5b`? In einem Satz: Welches executable ist größer und warum?

Abgabe: (Maximal) drei Sätze.

1.6 Optionale Zusatzaufgabe (keine Besprechung und ML)

Ganz interessant ist es, in die Assembler-Versionen (`.s`-Files) der Kompilate der C-Programme reinzuschauen.

Mit Hilfe des Tools `objdump` können außerdem weitere Einblicke in die Struktur der Binaries und in die Arbeitsweise des Linkers gewonnen werden. Bauen Sie z.B. zusätzlich in C-Programme Endlosschleifen ein und schauen Sie sich mit Hilfe der Process-ID die Inhalte der zugehörigen `proc`-Einträge an. Anhand der Adressen kann man auch Rückschlüsse auf Speicherorganisation, Seiten-Größen etc. gewinnen. Diese Elemente kommen im Verlauf der Vorlesung (etwas abstrakter) noch ausführlich zur Sprache.

2 Hausaufgabe (C-Programmierung)

Lernziele

Vertiefung C-Programmierung. (Klausuraufgabe aus dem letzten Jahr)

Aufgabe

Folgender Ausschnitt aus einem C Programm sei gegeben. Ersetzen sie die Platzhalter $\langle expr1 \rangle$, $\langle expr2 \rangle$, $\langle expr3 \rangle$ und $\langle expr4 \rangle$ so, dass das Programm eine doppelt verkettete Liste von 10 gleichartigen Personalakten erzeugt!

```
struct personalRecord{
    int salary;
    struct personalRecord* previousRecord;
    struct personalRecord* nextRecord;
};

int main(){
    int i;
    struct personalRecord firstRecord;
    struct personalRecord *theNewRecord;
    struct personalRecord *theRecordBeforeTheNewRecord;
    firstRecord.previousRecord = NULL;
    firstRecord.nextRecord = NULL;
    firstRecord.salary = 1000;
    theRecordBeforeTheNewRecord =  $\langle expr1 \rangle$  ;
    for(i=0; i<9; i++){
        theNewRecord = newRecord();
        theNewRecord->salary = 1000;
         $\langle expr2 \rangle$  = theRecordBeforeTheNewRecord;
         $\langle expr3 \rangle$  = theNewRecord;
        theRecordBeforeTheNewRecord = theNewRecord;
    }
    return 0;
}

struct personalRecord* newRecord(){
    struct personalRecord *newPersonalRecord;
    newPersonalRecord = (struct personalRecord *) malloc(  $\langle expr4 \rangle$  );
    return newPersonalRecord;
}
```

3 Tutoraufgabe (Quizfragen zu Tanenbaum)

Lernziele

Mit Hilfe der Fragen am Ende jedes Tanenbaum-Kapitels wird sichergestellt, dass man den Stoff soweit verinnerlicht hat, wie es das Ziel es jeweiligen Kapitels ist. Wir werden als Tutoraufgabe oft Fragen aus dem Tanenbaum erarbeiten.

Aufgabe

Beantworten Sie die folgenden Fragen zu Kapitel 1 (Englische Ausgabe von 2001):

- Frage 1: Was sind die 2 Hauptaufgaben eines Betriebssystems?
- Frage 2: Was ist Multiprogramming?
- Frage 3: Was ist Spooling? Hat es heute / in Zukunft noch Bedeutung?
- Frage 4: Auf Computersystemen ohne DMA: Welche Implikationen hat dies für Multiprogramming?
- Frage 8: Welche der folgenden Instruktionen sollte nur im Kernel-Mode ausgeführt werden?
 - Alle Interrupts diablen.
 - Die Zeit der System-Clock lesen
 - Die Zeit der System-Clock setzen
 - Die Memory Map ändern
- Frage 10: Eine CPU hat eine 4 stufige Pipeline. Jede Stufe benötigt 1 ns zur Ausführung. Wieviele Instruktionen pro Sekunde kann diese CPU ausführen? Was ändert sich bei einer 10-stufigen Pipeline?

Abgabe

Die Aufgabe wird in den Tutorübungen gemeinsam erarbeitet. Die Aufgabe soll NICHT abgegeben werden.