Tafelübung zu BS

0. Erste Schritte

Olaf Spinczyk
Arbeitsgruppe Eingebettete Systemsoftware

Lehrstuhl für Informatik 12
TU Dortmund

olaf.spinczyk@tu-dortmund.de
http://ess.cs.uni-dortmund.de/~os/

Agenda

- Organisatorisches
- Grundlagen C-Programmierung
- UNIX-Benutzerumgebung
  - Terminal, Shell
  - UNIX-Kommandos
  - GNU Compiler Collection (gcc)
- Aufgabe 0: Erste Schritte in C

Organisatorisches: Übungsaufgaben

- Theoriefragen und praktische Programmieraufgaben
- Vorstellung der neuen Aufgaben in der Tafelübung
- Bearbeitung in Dreiergruppen
  - Gruppenmitglieder sollten in derselben TÜ angemeldet sein
  - Lösungen werden komplett bewertet und die Punkte gleichermaßen an die Mitglieder verteilt
  - Hilfestellung:
    - betreute Rechnerübung! (Mi. 14:00-16:00, Do. 16:00-18:00, OH18, U01)
    - INPUD-Forum (http://inpud.cs.uni-dortmund.de/)
- Abgabe abhängig von Woche der Übung über ASSESS:
  - Gruppen 1,3,5,..: Donnerstag 12:00 bevor nächstes Blatt erscheint
  - Gruppen 2,4,6,..: Dienstag 12:00 nachdem das nächste Blatt erschienen ist
- Aufgabenblätter auf der Veranstaltungswebsite
- notwendig für erfolgreiche Übungsteilnahme: mindestens 50% der Punkte in *jeder* Aufgabe

Grundlagen C-Programmierung

- → Foliensatz C-Einführung (Folie 16-31)

UNIX-Benutzerumgebung

- Umgebung, Terminal, Shell
- UNIX-Kommandos
- GNU Compiler Collection (gcc)

Benutzerumgebung, Terminal

- diese Punkte machen (u.a.) einen UNIX-Account aus:
  - Benutzername
  - Identifikation (User-ID und Group-IDs)
  - Home-Directory
  - eingestellte Login-Shell
- Terminal
  - „Kommandozeile“
  - früher: dedizierte Endgeräte zur Kommunikation mit Zentral-rechner
  - heute: Terminal*emulation* (z.B. xterm, Konsole, gnome-terminal)

Quelle: http://de.wikipedia.org/w/index.php?title=Datei:Televideo925Terminal.jpg&filetimestamp=20060103074352

Terminal-Sonderzeichen

- einige Zeichen haben unter UNIX besondere Bedeutung
- Funktionen: u.a.
  - Korrektur von Tippfehlern
  - Steuerung der Bildschirm-Ausgaben
  - Einwirkung auf den Ablauf von Programmen
- Zuordnung Zeichen ↔ Funktion leider nicht einheitlich
- kann mit einem Kommando (**`stty(1)`**) verändert werden
- Übersicht:

<Backspace>	letztes Zeichen löschen
<Ctrl>-U	alle Zeichen der Zeile löschen
<Ctrl>-C	Interrupt – Programm abbrechen
<Ctrl>-Z	Stop – Programm wird angehalten
<Ctrl>-D	End Of File
<Ctrl>-S / <Ctrl>-Q	Bildschirmausgabe anhalten/fortsetzen

- auf deutschen Tastaturen: <Strg> statt <Ctrl>

UNIX-Kommandointerpreter: Shell

- meist stehen verschiedene Shells zur Verfügung: sh, ksh, csh, tcsh, bash, zsh...
- auf GNU-Systemen gebräuchlich: **bash**
- beim Öffnen eines Terminals startet die ausgewählte **Login-Shell**
- Wechsel der Login-Shell: **`chsh(1)`**

Aufbau eines UNIX-Kommandos

UNIX-Kommandos bestehen aus ...

- **Kommandoname** (der Name einer Datei, in der ein ausführbares Programm oder eine Kommandoprozedur für die Shell abgelegt ist)
  - nach dem Kommando wird automatisch in allen Verzeichnissen gesucht, die in der *Environment-Variable* $PATH gelistet sind
  - daher kann man normalerweise „**`ls`**“ schreiben statt „**`/bin/ls`**“
- einer Reihe von **Optionen** und **Argumenten**
  - Abtrennung Kommandoname/Optionen/Argumente durch Leerzeichen oder Tabulatoren
  - Optionen sind meist einzelne Buchstaben mit einem vorangestellten „-“ (Minuszeichen) (z.B. „**`ls -l`**“)
  - Argumente sind häufig Namen von Dateien, die von einem Kommando verarbeitet werden

UNIX-Kommandos

- man-Pages
- Dateisystem
- Benutzer
- Prozesse
- diverse Werkzeuge
- Texteditoren

man-Pages

- aufgeteilt in verschiedene *Sections*
  (1) Kommandos
  (2) Systemaufrufe
  (3) C-Bibliotheksfunktionen
  (5) Dateiformate (spezielle Datenstrukturen etc.)
  (7) Verschiedenes (z.B. IP, GPL, Zeichensätze, ...)
- man-Pages werden normalerweise mit der Section zitiert: **`printf(3)`**
  - sonst teilweise mehrdeutig (**`printf(1)`** vs. **`printf(3)`**)!
- Aufruf unter Linux:

```
man [section] Begriff
z.B.
hsc@uran:~$ man 3 printf
```

- Suche nach Sections: **`man -f Begriff`**
- Suche nach Stichwort: **`man -k Stichwort`**
- mehr Informationen über man: **`man man`**

Dateisystem

ls		Verzeichnis auflisten; wichtige Optionen:
	-l	langes Ausgabeformat
	-a	auch mit . beginnende Dateien werden gelistet
pwd		Aktuelles Verzeichnis ausgeben
chmod		Zugriffsrechte einer Datei ändern
cp		Datei(en) kopieren
mv		Datei(en) verlagern (oder umbenennen)
ln		Datei linken (weiteren Verweis auf dieselbe Datei erzeugen)
ln -s		symbolischen Link erzeugen
rm		Datei(en) löschen
mkdir		Verzeichnis erzeugen
rmdir		Verzeichnis löschen (muss leer sein!)

Benutzer

id, groups	eigene Benutzer-ID, Gruppenzugehörigkeit
who	am Rechner angemeldete Benutzer

Prozesse

ps	Prozessliste ausgeben
-u x	Prozesse des Benutzers x
-ef	alle Prozesse (-e), ausführliches Format (-f)
top -o cpu	Prozessliste, sortiert nach aktueller Aktivität
kill <*pid*>	Prozess „abschießen“ (Prozess kann aber dennoch geordnet terminieren oder sogar ignorieren)
kill -9 <*pid*>	Prozess „gnadenlos abschießen“ (Prozess kann nicht mehr hinter sich aufräumen oder ignorieren)

diverse Werkzeuge

cat	Dateien hintereinander ausgeben
more, less	Dateien bildschirmweise ausgeben
head, tail	Anfang/Ende einer Datei ausgeben (10 Zeilen)
pr, lp, lpr	Datei ausdrucken
wc	Zeilen, Wörter und Zeichen zählen
grep, fgrep, egrep	nach bestimmten Mustern o. Wörtern suchen
find	Dateibaum durchlaufen
sed	Stream-Editor, z.B. zum Suchen/Ersetzen
tr	Zeichen aufeinander abbilden oder löschen
awk	Pattern-Scanner
cut	einzelne Felder aus Zeilen ausschneiden
sort	sortieren

Texteditoren

- Geschmackssache – aber einen solltet ihr beherrschen!
- Klassiker mit Nerdfaktor: **vim, emacs**
  (mit graphischem Frontend: **xemacs, gvim**)
- Minimalisten: **pico, nano**
- weitere mit X-Frontend: **kwrite, kate, gedit, Eclipse, ...**
- zum Programmieren **nicht geeignet**:
  Office-Programme (MS Word, OpenOffice Writer, ...)

GNU Compiler Collection

- ursprünglich: *GNU C Compiler*
- mittlerweile: Sammlung von verschiedenen Compilern (u.a. C, C++, Java, Objective-C, Fortran 95, ...)
- viele versch. Zielplattformen (x86, AMD64, Alpha, IA-64 ...)
- C-Compiler: **gcc**
- Compilieren und Linken eines C-Programms:

```
gcc -Wall -o sum_n sum_n.c
```

-Wall alle Warnungen ausgeben
-o <*Ziel*> Name für ausführbare Datei

- weitere nützliche Parameter (siehe man-Page):
  -Werror, -ansi, -pedantic, -D_POSIX_SOURCE
- Warnungen sind grundsätzlich ernstzunehmen und zu beseitigen, daher möglichst immer mit **-Werror** übersetzen!

Aufgabe 0: Erste Schritte in C

```
/* sum_n.c: Addiert alle Zahlen von 1 bis n */
#include <stdio.h>
int sum_n(int n) {
   int i = 1, res = 0;
   while (i <= n) res += i++;
   return res;
}
int main() {
   printf("%d\n", sum_(5));
   return 0;
}
```

- compilieren/linken:

```
hsc@uran:~/bs/a0$ gcc -Wall -o sum_n sum_n.c
sum_n.c: In function 'main':
sum_n.c:9: warning: implicit declaration of function 'sum_'
/tmp/ccKCnWMh.o: In function `main':
sum_n.c:(.text+0x3e): undefined reference to `sum_'
collect2: ld returned 1 exit status
hsc@uran:~/bs/a0$
```

- Da haben wir uns wohl vertippt ...

Aufgabe 0: Erste Schritte in C

```
/* sum_n.c: Addiert alle Zahlen von 1 bis n */
#include <stdio.h>
int sum_n(int n) {
   int i = 1, res = 0;
   while (i <= n) res += i++;
   return res;
}
int main() {
   printf("%d\n", sum_n(5));
   return 0;
}
```

- compilieren/linken:

```
hsc@uran:~/bs/a0$ gcc -Wall -o sum_n sum_n.c
hsc@uran:~/bs/a0$ ls
sum_n    sum_n.c
```

- ausführen:

```
hsc@uran:~/bs/a0$ ./sum_n
15
hsc@uran:~/bs/a0$
```

Aufgabe 0: Erste Schritte in C

- mit UNIX-Umgebung experimentieren
  - in der Rechnerübung,
  - in der Linux-VM von der BS-Website, oder
  - in einer eigenen Linux-Installation
- Adressen von Variablen ausgeben:
  - Adressoperator: &
  - Format für printf(): %p

```
#include <stdio.h>
int main(void) {
    int x;
    printf("Die Variable x ist an Adresse %p.\n", (void *)&x);
    return 0;
}
```