Hochschule RheinMain
Fachbereich DCSM - Informatik
Prof. Dr. Robert Kaiser
Sebastian Flothow
Alexander Schönborn
Daniel Schultz

Betriebssysteme

WS 2020/21
LV 3122

Aufgabenblatt 1
Bearbeitungszeit 2 Wochen
Abgabetermin: 23.11.2020, 4:00 Uhr

Aufgabe 1.1 (Repository):

In diesem Praktikum werden die Abgaben in einem zentralen SVN-Repository verwaltet. Die Praktikumsleiter legen dazu für jedes Aufgabenblatt eine Vorlage an, die Sie am Anfang der Praktikumsveranstaltung auschecken müssen. Machen Sie sich mit der Handhabung von SVN vertraut. Weiterführende Hilfe finden Sie dazu im Wiki des Studiengangs und im Internet, z.B.:

`https://www-intern.cs.hs-rm.de/publicwiki/index.php/Hauptseite`

In den Vorlagen finden Sie eine einheitliche Projektstruktur vor, die Sie nicht verändern dürfen. Das Projekt beinhaltet in der Regel Templates für alle Quelldateien und die anzulegende Dokumentation, sowie ein Makefile, um das Projekt mit den gewünschten Compiler-Einstellungen unter Linux zu kompilieren:

`https://svn.cs.hs-rm.de/svn/bs20_vnnnn001/1/`	Vorlage Aufgabenblatt 1
`https://svn.cs.hs-rm.de/svn/bs20_vnnnn001/2/`	Vorlage Aufgabenblatt 2
`https://svn.cs.hs-rm.de/svn/bs20_vnnnn001/3/`	Vorlage Aufgabenblatt 3
...	...

Initialer Checkout der Aufgaben aus dem Repository (mit Ihrer Nutzerkennung):

`$ svn checkout https://svn.cs.hs-rm.de/svn/bs20_`**`vnnnn001`**`/`

Aktualisierung des Repositorys:

```
$ svn update
```

Anzeigen der lokal geänderten Dateien:

```
$ svn status
```

Anzeigen der lokalen Änderungen:

```
$ svn diff
```

Speichern der lokalen Änderungen im Repository (wichtig!):

```
$ svn commit
```

Bei Problemen mit dem SVN-Zugang wenden Sie sich bitte an die Laboringenieure.

Aufgabe 1.2 (Bedienung SVN):

Öffnen Sie die Datei `mybcp.c` im Editor Ihrer Wahl und ändern Sie die Zeile

```
printf("Hallo, Praktikum!\n");
```

zu

```
printf("Hallo, Betriebssysteme-Praktikum!\n");
```

um. Speichern Sie die Änderungen in der Datei ab. Schauen Sie sich die lokalen Änderungen an (`svn status`, `svn diff`) und committen Sie die Änderung ins SVN-Reposity (`svn commit`).

Aufgabe 1.3 (C-Programmierung):

In diesem Praktikum wird in C programmiert. Auch wenn Sie bisher noch nicht in C programmiert haben, werden Sie die Grundlagen von C schnell beherrschen. Die Aufgaben sind so ausgelegt, dass Sie sie auch mit Ihren Java-Kenntnissen aus den bisherigen Semestern lösen können. Im Internet finden Sie zahlreiche Bücher und Online-Tutorials zum Thema C-Programmierung:

J. Gusted, *Modern C*: `https://modernc.gforge.inria.fr/`

J. Wolf, *C von A bis Z*: `http://openbook.rheinwerk-verlag.de/c_von_a_bis_z/`

Im weiteren Verlauf des Praktikums wird davon ausgegangen, dass Sie sich einen auch für andere Personen *lesbaren* Programmierstil angewöhnen sollten. Dies bedeutet insbesondere, dass Sie auf Einrückungen und die richtige Formatierung Ihres Quellcodes achten sollten. Setzen Sie alle Anweisungenen nach `if/else/for/while/...` am besten immer in geschweifte Klammern. Trennen Sie inhaltlich nicht zusammenhängende Zeilen durch Leerzeilen. Als Leitfaden aus der Praxis für gut lesbaren Code gilt z.B. der *Linux Kernel Coding Style*:

`https://www.kernel.org/doc/html/v4.10/process/coding-style.html`

Aufgabe 1.4 (UNIX-Kommandos):

Wiederholen Sie den Umgang mit den allgemeinen UNIX-Kommandos. Im Weiteren wird die Kenntnis des praktischen Umgangs mit dem UNIX-System auf Kommandoebene als bekannt vorausgesetzt. Eine gute Einführung in das Thema bietet unter anderem das *SelfLinux* Tutorial: `http://www.selflinux.org/selflinux/`

Machen Sie sich weiterhin mit den sogenannten Manual-Pages in einem Unix-System vertraut. Die Manual-Pages sind in Gruppen organisiert. Gruppe 1 behandelt UNIX Kommandos, Gruppe 2 die Systemdienstschnittstelle, Gruppe 3 die Dokumentation der C-Bibliotheken, etc. Der Befehl `apropos` auf der Kommandozeile zeigt für ein Stichwort die vorhandenen Manual-Pages an. Das Kommando `man` zeigt die jeweilige Manual-Page an:

```
$ apropos printf
$ man 3 printf
```

Unter dem Stichwort „Systemprogrammierung“ finden Sie ebenfalls viel Material im Internet, wie zum Beispiel J. Wolf, *Linux-UNIX-Programmierung*, welches kostenlos verfügbar ist:

`http://openbook.rheinwerk-verlag.de/linux_unix_programmierung/`

Aufgabe 1.5 (Fehlerbehandlung):

Beachten Sie, dass UNIX-Systemaufrufe den Status der ausgeführten Operation über ihren Rückgabewert zurückliefern. In der Regel wird eine erfolgreiche Operation durch die Zahl 0 oder einen positiven Integer angezeigt. Bei Fehlern wird dagegen der Wert -1 zurückgeliefert. Weitergehende Fehlermeldungen können mittels der Systemvariable `errno` bzw. über die Hilfsfunktion `perror()` ausgegeben werden:

```
...
fd = open("dateiname", O_RDONLY);
if (fd == -1) {
    perror("Fehler bei open");
    exit(EXIT_FAILURE);
}
...
```

Die Fehlerprüfungen sind wichtig für das Praktikum. Sie sollten nach *jedem* Systemaufruf durchgeführt werden!

Aufgabe 1.6 (Ungepufferte Dateiein-/ausgabe):

In dieser Aufgabe wird der Umgang mit Dateien mittels UNIX-Systemaufrufen geübt. Unterscheiden Sie zwischen Systemaufrufen und vergleichbaren Bibliotheksfunktionen der ANSI C Standard-I/O-Bibliothek. Beachten Sie die notwendigen Header-Dateien. Zur Beschreibung der Systemaufrufe sei auf die Manual-Pages, Kapitel 2, verwiesen. Beachten Sie, dass Systemaufrufe in der Regel mit den Rückgabeparametern auch Fehler anzeigen. Diese müssen überprüft werden, ansonsten wird das Programm in der Beurteilung durch den Praktikumsleiter abgewertet.

`int open(const char *name, int oflag);` bzw. `int open(const char *name, int oflag, mode_t mode);`	Öffnen einer Datei
`int creat(const char *name, mode_t mode);` analog `int open(name, O_WRONLY|O_CREAT|O_TRUNC, mode)`	Erzeugen einer neuen Datei
`int close(int fd);`	Schließen einer Datei
`ssize_t read(int fd, void *buf, size_t nbytes);`	Lesen aus einer Datei
`ssize_t write(int fd, const void *buf, size_t nbytes);`	Schreiben in eine Datei
`off_t lseek(int fd, off_t offset, int whence);`	Positionieren in einer Datei

a) Schreiben Sie ein C-Programm `mybcp.c` , das eine beliebige Datei byteweise kopiert. Der Name der Quelldatei und der Name der Zieldatei sollen beim Programmaufruf über die Kommandozeile übergeben werden, d.h. ein Aufruf des Programms lautet `mybcp` *`fromfile tofile`*. Zunächst soll die erzeugte Datei Lese- und Schreibrecht für den Eigentümer besitzen, keine Rechte für alle anderen (`rw-------`).

b) Schreiben Sie ein Programm `mybappend.c`, das den Inhalt einer Datei byteweise an eine bestehende Datei anfügt. Die Namen der Ausgangs-/Zieldatei sowie der anzufügenden Datei sollen wieder beim Programmaufruf über die Kommandozeile übergeben werden.

c) Modifizieren Sie Ihr Programm aus (a) so zu einem Programm `myrevbcp.c`, dass die erzeugte Datei die Folge der Bytes in umgekehrter Reihenfolge enthält (`lseek()` verwenden). *Hinweis:* Sie können Ihre Implementierung prüfen, in dem Sie eine Datei *zweimal*

umkehren und das Endergebnis mit der Ursprungsdatei mit Hilfe des Dienstprogramms `diff` vergleichen.

d) Modifizieren Sie Ihr Programm aus (a) so zu einem Programm `mycp.c`, dass die Puffergröße in einem `read()` bzw. `write()` Systemaufruf über die Kommandozeile wählbar ist (Aufruf `mycp` *`fromfile tofile buffersize`*).

Hinweis zum Testen: Sie können die Ergebnisse Ihrer Programme anhand von Test-Skripten prüfen. Die Skripte sind in der Regel nach dem jeweiligen Programm benannt, z.B. `test_`*`mybcp`*`.sh` für *mybcp*. Mit folgendem Kommando können Sie in einer Zeile Ihr Programms kompilieren und die Tests ausführen:

```
studi@hsrm~/bs/1 $ make && ./test_mybcp.sh
cc -std=c11 [...] -o mybcp mybcp.c
Kopiere 'datei1' zu 'datei2'
---[Ausgaben des Programms]-------------
Kopiere Datei byteweise ...
----------------------------------------
Vergleiche Groesse der Dateien: OK
Vergleiche Inhalt der Dateien: OK
Teste Zugriffsrechte der Datei: OK
studi@hsrm ~/bs/1 $
```

Aufgabe 1.7 (Einfache Dateiattribute):

`int stat(const char *name, struct stat *buf);` bzw. `int fstat(int fd, struct stat *buf);`	Ermitteln von Dateiattributen
`int truncate(const char *name, off_t length);` bzw. `int ftruncate(int fd, off_t length);`	Setzen der Dateilänge

a) Schreiben Sie ein C-Programm `filelength.c`, das die Länge einer Datei ausgibt, deren Namen über die Kommandozeile übergeben wird. Vergleichen Sie Ihre Ausgabe mit der Ausgabe des Dienstprogramms `ls -l`.

b) Schreiben Sie ein C-Programm `grow.c`, das die Länge einer Datei auf die angegebene Größe setzt. Der Name der Datei sowie die Dateilänge werden über die Kommandozeile übergeben (Aufruf `grow` *`bigfile length`*). Falls die Datei nicht existiert, soll sie mit Lese- und Schreibrechten für den Eigentümer neu angelegt werden.

c) Modifizieren Sie Ihr Programm `mycp.c` so, dass die Rechtefestlegungen von Originaldatei und kopierter Datei identisch sind.

d) Modifizieren Sie Ihr Programm `mycp.c` aus (c) so, dass eine entsprechende Fehlerausgabe erfolgt, wenn es nicht auf reguläre Dateien angewendet wird.

Vergessen Sie nicht, Ihre geänderten Dateien im Projekt zu committen!