BERGISCHE UNIVERSITÄT
GESAMTHOCHSCHULE WUPPERTAL
GAUSS-STRASSE 20
42097 WUPPERTAL
(Korrespondenzanschrift)
42119 WUPPERTAL
(Lieferanschrift)
TELEX 8 592 262 bughw
TELEFAX (0202) 439-2901
TELEFON (0202) 439-1

Fachbereich 7

MATHEMATIK

Prof. Dr. Hans-Jürgen Buhl
Praktische Informatik / Numerik

e-mail: Juergen.Buhl@math.uni-wuppertal.de

Betriebssysteme: Konzepte, Dienste, Schnittstellen (Betriebssysteme und betriebssystemnahe Programmierung)

SS 2003 – Übungsblatt 6

25. Juni 2003
Ausgabe: 18. Juni 2003

Aufgabe 1. *Hochfahren eines Linux-Systems*

Beschreiben Sie in eigenen Worten die verschiedenen Phasen beim Booten eines Linux-Systems.

Aufgabe 2. *cat1.c*

Bringen Sie das folgende Programm zum Ablauf

```
/* cat1.c -- simple version of cat */
#include <stdio.h>
#include <unistd.h>

/* While there is data on standard in (fd 0), copy it to standard
   out (fd 1). Exit once no more data is available. */

int main(void) {
    char buf[1024];
    int len;

    /* len will be >= 0 while data is available, and read() is
       successful */
    while ((len = read(STDIN_FILENO, buf, sizeof(buf))) > 0) {
        if (write(STDOUT_FILENO, buf, len) != len) {
            perror("write");
            return 1;
        }
    }

    /* len was <= 0; If len = 0, no more data is available.
```

```
       Otherwise, an error occurred. */
    if (len < 0) {
        perror("read");
        return 1;
    }

    return 0;
}
```

Lesen Sie die Manual-Seiten von `read` und `write`. Erklären Sie die Wirkungsweise des Programms. Ist das Programm korrekt?

Aufgabe 3. *cat2.c*

Bringen Sie das folgende Programm zum Ablauf

```
/* cat2.c -- simple 2nd version of cat */
#include <stdio.h>
#include <unistd.h>

/* While there is data on standard in (fd 0), copy it to standard
   out (fd 1). Exit once no more data is available. */

int main(void) {
    char buf[1024];
    int len;
    int wlen, n;

    /* len will be >= 0 while data is available, and read() is
       successful */
    while ((len = read(STDIN_FILENO, buf, sizeof(buf))) > 0) {
        if (len == -1){
            perror("write");
            return 1;
        }
        wlen = 0;
        do {
            if ((n = write(STDOUT_FILENO, &buf[wlen], len-wlen)) == -1) {
                perror("write");
                return 1;
            }
            wlen += n;
        } while (wlen < len);
    }

    /* len was <= 0; If len = 0, no more data is available.
       Otherwise, an error occurred. */
    if (len < 0) {
        perror("read");
        return 1;
```

```
    }

    return 0;
}
```

und erklären Sie seine Wirkungsweise Zeile für Zeile. Warum unterscheidet es sich von `cat1.c`?

Aufgabe 4. *cat.c aus coreutils*

Installieren Sie auf Ihrem Linux-System die Quellen der `coreutils`, die unter anderem `cat.c` des Unix-Betriebssystems enthalten (benutze anschließend `rpm -bp coreutils.spec`).

Betrachten Sie dann im Verzeichnis

`/usr/src/packages/BUILD/coreutils-4.5.8/src`

die Datei `cat.c`. Welche zusätzlichen Funktionalitäten enthält sie?

Übersetzen Sie (als Superuser):

```
cd /usr/src/packages/BUILD/coreutils-4.5.8
./configure
make all
file src/cat

make check

make clean

make distclean
```

Was läuft in den einzelnen Phasen ab?

Aufgabe 5. *net-tools*

Besorgen Sie sich analog die `net-tools`. Entpacken Sie und übersetzen Sie mittels:

```
cd /usr/src/packages/BUILD/net-tools-1.60
make config
make

./hostname

make clean
```

Versuchen Sie den Inhalt von `hostname.c` in groben Zügen funktional zu beschreiben.