Teaching Assistants:
Adrian Kunz, Lukas Kawollek, Yonas Woldetsadik, Gabriel Selle, Tom Reuter

Blatt 2

Besprechung: 06.11.2025, 11:00 Uhr

Aufgabe 2.1: Programmieraufgabe: Interaktion mit dem Betriebssystem über Systemcalls

$1+1=$ *2 Punkte*

a) Begründen Sie die Unterteilung in User Mode und Kernel Mode.

b) Welche der `xv6` System Calls sind ebenfalls in Linux vorhanden, bzw. welche fehlen? Haben diese den gleichen oder einen unterschiedlichen Namen? Für die Liste der Linux System Calls können Sie beispielweise Wikipedia verwenden.[1]

c) Betrachten Sie Abschnitt 5 des Unix Papers[2] aus dem Jahr 1974. Welche der dort genannten System Calls finden sich in xv6 oder Linux?

(1 Punkt)

d) Schreiben Sie ein Programm in C, welches mindestens folgende Aufrufe bezüglich Datei- und Verzeichnissystemverwaltung in der main-Funktion ausführt:

(a) Verzeichnis erzeugen
(b) Datei im neuen Verzeichnis anlegen
(c) Neue Datei öffnen und mit “123” beschreiben
(d) Datei schließen

e) Nennen Sie vier Systemaufrufe, die Ihr Programm für die Dateioperationen ausführt. Hinweis: Sie können die System Calls durch `strace ./ihrProgramm` anzeigen lassen.

(1 Punkt)

[1] `https://de.wikipedia.org/wiki/Liste_der_Linux-Systemaufrufe`
[2] `https://dsf.berkeley.edu/cs262/unix.pdf`

Aufgabe 2.2: Prozesse in Linux

1 Punkt

Gegeben sei der nachfolgende Code eines Prozesses A, der zum Zeitpunkt $t = 0$ gestartet wird. Skizzieren Sie den zeitlichen Ablauf der Ausführung als Diagramm und stellen Sie dabei die Verwandtschaftsbeziehungen der Prozesse dar.

```c
#include <sys/types.h>

void main(void)
{
    int i;
    sleep(10);
    fork();
    sleep(10);
    i = fork();
    sleep(20);
    if (i != 0)
        waitpid(i, NULL);
    sleep(30);
    fork();
    sleep(0);
}
```

Hinweis 1: Beachten Sie die spezielle Eigenschaft von `fork()`!

Hinweis 2: Sollten Sie versuchen, den abgebildeten Code zu implementieren, wird dieser vermutlich nicht ohne Anpassung lauffähig sein. Für die Lösung der Aufgabe ist keine Implementierung notwendig.

Hinweis 3: Für die Funktionen der Prozess-API, siehe das Unix Paper.

Aufgabe 2.3: Prozesstabelle

1 Punkt

In der Vorlesung wurde erläutert, dass ein Betriebssystem alle Prozesse in einer Liste, bzw. Tabelle verwaltet.

a) Welche Informationen sollten dazu in den so genannten Prozesskontrollblöcken (PCB – Process Control Block) für jeden Prozess gespeichert werden? Betrachten Sie den PCB in `xv6` (`struct proc` in `kernel/proc.h`) und nennen Sie mindestens sechs Elemente und erklären Sie diese.

b) Wird in einem System, in dem die Rechenzeit zwischen den Prozessen aufgeteilt wird (Time-Sharing-System), eine Prozesstabelle (Tabelle welche die PCBs enthält) benötigt? (In `xv6` ist diese Tabelle als `struct proc proc[NPROC]` Array in `kernel/proc.c` zu finden)

c) Wird die Tabelle bei Personalcomputern benötigt, bei denen jederzeit nur ein Prozess existiert und die gesamte Maschine für seine Ausführungszeit belegt ist?

Aufgabe 2.4: Interprozesskommunikation mit Signalen

$1+1=2$ *Punkte*

Machen Sie sich mit der Interprozesskommunikation über Signale vertraut[3].

a) Beschreiben Sie kurz die Funktionsweise des nachfolgenden Programms.

b) Übersetzen Sie das nachfolge Programm (`gcc -o signal signal.c` und starten es in einem Terminal. Nutzen Sie ein zweites Terminal, um mit den Kommandos `ps` und `kill` die Ausgabe "Die Konfiguration wurde neu geladen" zu erzeugen.

(1 Punkt)

c) Erweitern Sie das nachfolgende Programm so, dass auch das Signal SIGINT behandelt werden kann. Übersetzen Sie das Programm neu, starten es und drücken Ctrl + C. Was passiert?

(1 Punkt)

```c
#include<stdio.h>
#include<signal.h>
#include<unistd.h>
#include<stdlib.h>

void sig_handler(int signo)
{
    if (signo == SIGUSR1)
        printf("Die Konfiguration wurde neu geladen.\n");
}

int main(void)
{
    if (signal(SIGUSR1, sig_handler) == SIG_ERR)
        printf("Ich kann kein Signal empfangen\n");

    while(1)
        sleep(1);
    return 0;
}
```

[3] https://openbook.rheinwerk-verlag.de/linux_unix_programmierung/Kap08-000.htm

Aufgabe 2.5: Interprozesskommunikation mit Pipes

$1+1=2$ *Punkte*

a) Beschreiben Sie kurz das Konzept einer Pipe anhand folgendem Aufruf in einer Shell: `ps | wc -l`.

b) Wann eignet sich die Verwendung einer Pipe, gegenüber der Nutzung von Signalen?

(1 Punkt)

c) Nachfolgend wird in einem C-Programm das Konzept der Pipe genutzt. Welche Prozesse werden erstellt und welcher Prozess kommuniziert hier mit wem und wie wird die Pipe dabei genutzt?

(1 Punkt)

```c
#include <sys/wait.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>

int main(int argc, char *argv[]) {
    pid_t cpid = fork();
    if (cpid == -1) {
        fprintf(stderr, "Kind Fork Fehler\n");
        exit(EXIT_FAILURE);
    }
    if (cpid == 0) {
        int pipefd[2]; char buf;

        if (pipe(pipefd) == -1) {
            fprintf(stderr, "Pipe Fehler\n");
            exit(EXIT_FAILURE);
        }
        pid_t gcpid = fork();
        if (gcpid == -1) {
            fprintf(stderr, "Enkel Fork Fehler\n");
            exit(EXIT_FAILURE);
        }
        if (gcpid == 0) {
            close(pipefd[1]);

            while (read(pipefd[0], &buf, 1) > 0)
                write(STDOUT_FILENO, &buf, 1);

            write(STDOUT_FILENO, "\n", 1);
            close(pipefd[0]);
            _exit(EXIT_SUCCESS);
        } else {
            char *text = "Hallo, ich bins!\n";
            close(pipefd[0]);
            write(pipefd[1], text, strlen(text));
            close(pipefd[1]);
            wait(NULL);
            exit(EXIT_SUCCESS);
        }
    } else {
        wait(NULL);          // Warte auf Terminierung des Kindes
        exit(EXIT_SUCCESS);
    }
}
```

Abgabe bis zum 06.11.2025 bis 10:00 Uhr

- Die Voraussetzung für die Bewertung ist die Abgabe der Lösung via Moodle (`https://moodle.uni-kassel.de/course/view.php?id=19269`).
- Erlaubte Dateianhänge:
  - PDF, TXT für Dokumente
  - Quellcode
  - JPG, PNG für Grafiken und Bilder
  - Komprimiert im ZIP-Format (kein RAR!).
- Lösungen, die diesen Kriterien nicht entsprechen, werden nicht berücksichtigt!
- In diesem Übungsblatt gibt es insgesamt **8 Punkte**. Davon müssen mindestens **4 Punkte** erreicht werden.