Professur für Betriebssysteme und Verteilte Systeme
Wintersemester 2025/2026, Prof. Dr. Bettina Schnor

Übung zur Vorlesung
Grundlagen Betriebssysteme und Rechnernetze

1. Übungsblatt, Abgabe bis Mo, 03. November 2025, 10:00 Uhr

Hinweise zur Bearbeitung und Abgabe der Übung:

- Die Übungsaufgaben und Zusatzmaterial finden Sie auf Moodle: `https://moodle2.uni-potsdam.de/course/view.php?id=42684`
- Abgabe in 3er-Gruppen; tragen Sie sich im Moodle-Kurs in eine Abgabegruppe ein! Auf allen Abgaben Namen und Matrikelnummern der Gruppenmitglieder nicht vergessen!
- Laden Sie Ihre Lösungen der Theorieaufgaben als PDF in Moodle hoch.
- Nutzen Sie für die Praxisaufgaben Git.UP! Klonen Sie sich für die jeweilige Aufgabe die Vorlage aus der Gruppe `GBR Vorlagen`. Sollte es für die Aufgabe keine Vorlage geben, zB: Aufgabe 1.1, erstellen Sie ein eigenes Repository. Achten Sie darauf ihre Git-Projekte im GBR Namespace zu erstellen. Für weitere Details folgen sie dem Manual auf Moodle.
- Halten Sie sich an die in den praktischen Aufgaben gegebenen Dateinamen und Methodennamen!
- Bitte keine Umlaute und Leerzeichen in Verzeichnissen und Dateinamen in der Abgabe verwenden! Ersetzen Sie Umlaute in Dateinamen (ä wird zu ae etc.)!
- Bei Problemen mit den Praxisaufgaben oder den Abgaben in GitLab legen Sie bitte in Ihrem GitLab-Repository ein Issue an und markieren darin einen Lehrenden! Bei Fragen zu den Aufgaben benutzen Sie bitte das Diskussionsforum in Moodle!
- Bitte kommentieren Sie Ihre Programme sinnvoll! Verwenden Sie sprechende Bezeichner und behandeln Sie mögliche Fehler!
- Stellen Sie sicher, dass Ihre Programme auf `tiree.lab.cs.uni-potsdam.de` compilierbar und lauffähig sind!

Gesamtpunktzahl des Aufgabenblattes: 20 Punkte.

Aufgabe 1.1: Sizeof und Arrays

Gegeben sei das folgende C-Code-Fragment:

```
int a[8];
int i;

for (i = 0; i < sizeof(a); i++) {
    a[i] = 1;
}
```

Bearbeiten Sie die folgenden Teilaufgaben:

a) Geben Sie den Speicherplatz für das **gesamte Array a** an, vorausgesetzt eine einzelne Integer-Variable benötigt 4 Byte Speicherplatz. Bestimmen Sie zudem den Wert, den der **sizeof**-Operator für **a** zurückgibt.

b) Mit Hilfe der Schleife sollen **nur** die Elemente des Arrays **a** auf 1 gesetzt werden. Erläutern Sie, ob die Schleife die gewünschte **Funktionalität erfüllt**. Erklären Sie, ob **genau** die benötigte Anzahl von Elementen auf 1 gesetzt werden. Passen Sie außerdem das gegebene Programm an, sodass **genau** die Elemente von **a** von der Schleife durchlaufen werden.

c) Ermitteln Sie den Wert des **sizeof** Operators für **a** bei dem folgenden Programm:

```
unsigned long size(int* a){
    return sizeof(a);
}
int main(int argc, char** argv){
    int a[8];
    printf("%lu",size(a));
}
```

d) Erläutern Sie, wie das Programm angepasst werden muss, damit innerhalb der **Funktion size**, die Länge des **Arrays a** ermittelt werden kann.

(0.5+1+0.5+1 Punkte)

Aufgabe 1.2: Prozesszustände

Gegeben sei ein System mit **einem CPU-Kern**. Das System hat Zugriff auf eine Festplatte und einen USB-Datenträger und nutzt für jedes Gerät eine **eigene Warteschlange**.
Die Latenz für den Zugriff auf die Festplatte beträgt **35 ms** und die des USB-Datenträgers beträgt **45 ms**.
Zum Start sind die folgenden 3 Prozesse A, B und C mit einer Laufzeit von **50ms** im

Zustand „ready“ in der Ready-Queue **alphabetisch aufsteigend** einsortiert.
Für diese Prozesse treten die folgenden Ereignisse nacheinander auf. Dabei unterbricht das System nach **10 ms** den laufenden Prozess und der Scheduler wählt einen neuen Prozess aus der Ready-Queue aus. **Vernachlässigen** Sie im Rahmen der Übungsaufgabe den Zeitaufwand des Schedulers und des Dispatchers.

i) Geben Sie die Zustände der **Prozesse und der Warteschlangen** nach jedem Ereignis mithilfe einer Tabelle an. Nutzen Sie dafür das Zustandsmodell mit **5 Zuständen** für Prozesse aus der Vorlesung.

   1. Nachdem der Prozess B **5ms** arbeitet, greift er auf den USB-Datenträger zu.
   2. Nachdem der Prozess A **15ms** arbeitet, greift er auf die Festplatte zu.
   3. Nachdem der Prozess A **20ms** arbeitet, greift er auf den USB-Datenträger zu.

ii) Erklären Sie, welchen Vorteil es hat, dass eine Warteschlange **pro Gerät**, anstatt einer Warteschlange für alle Geräte, verwendet wird.

Hinweis: Wenn 2 Prozesse gleichzeitig eintreffen, werden diese in die Warteschlange alphabetisch aufsteigend einsortiert.

(3+1 Punkte)

Aufgabe 1.3: Prozessverwaltung

Die Kommando `ps` kann unter Linux zu der folgenden Ausgabe führen (Auszug):

```
USER          PID    PPID %CPU %MEM TT       STAT  STARTED     TIME CMD
root          956       1  0.0  0.0 ?        Ss    01:41:44 00:00:00 login -- mschroetter
mschroe+      973       1  0.0  0.0 ?        Ss    01:41:49 00:00:00 /usr/lib/systemd/
    ↪ systemd --user
mschroe+     1009     956  1.8  0.1 tty2     Ss    01:41:49 00:00:13 -zsh
mschroe+     1096     973  0.0  0.0 ?        S<sl  01:41:50 00:00:00 /usr/bin/pipewire
mschroe+     1097     973  0.0  0.1 ?        S<sl  01:41:50 00:00:00 /usr/bin/wireplumber
mschroe+     1929    1009  0.4  0.0 ?        Ss    01:53:56 00:00:01 tmux new-session
mschroe+     3533    1929  3.3  0.1 pts/3    Ss    01:58:25 00:00:00 -zsh
mschroe+     3643    3533  0.0  0.0 pts/3    S+    01:58:32 00:00:00 nvim test.c
mschroe+     3644    3643  5.2  0.5 ?        Ssl   01:58:32 00:00:00 nvim --embed test.c
mschroe+     3676    3644 20.8  3.6 ?        Ssl   01:58:32 00:00:03 node language-server.
    ↪ js --stdio
mschroe+     3688    3644  1.0  0.2 ?        Ssl   01:58:33 00:00:00 clangd
mschroe+     3722    1929  1.7  0.1 pts/4    Ss    01:58:34 00:00:00 -zsh
mschroe+     3787    3722  1.7  0.1 pts/4    S+    01:58:35 00:00:00 neomutt
mschroe+     3828    1929  5.7  0.1 pts/5    Ss    01:58:37 00:00:00 -zsh
mschroe+     3967    3828  100  0.0 pts/5    R+    01:58:48 00:00:00 ps -eo user,pid,ppid
    ↪ ,%cpu,%mem,tty,stat,start,time,cmd
```

a) Erläutern Sie die jeweiligen Spaltenüberschriften!

b) In welchem Zustand ist der Prozess mit der PID 3787?

c) Geben Sie für den Prozess mit der PID 3967 das Prozessabstammungsdiagramm an! (3 Punkte)

Aufgabe 1.4: CoFee

CoFee hilft Ihnen Fehler in Ihren Programmen zu finden und zu beheben. In der Vorlage **Uebung 1 CoFee** auf **Git.up** finden Sie 2 C-Programme. Analysieren Sie beide Programme mithilfe von CoFee. **Beheben** Sie die von CoFee gemeldeten Fehler in Ihrem Repository.

(3 Punkte)

Aufgabe 1.5: Prozesstabelle

Implementieren Sie eine Prozesstabelle mithilfe einer **doppelt** verketteten Liste.

Die Elemente der Liste entsprechen den vereinfachten Prozessleitblöcken. Für jeden Prozess umfasst der zugehörige Leitblock die Prozess-ID (**PID**) und die ID des Elternprozesses (parent PID, **PPID**).

Die Datei **dl_proc_list.c** umfasst nach Ihrer Implementation mindestens den folgenden Datentypen sowie die Funktionen:

- Den Datentyp **dl_proc_list**,
- die Funktion **dl_proc_list* dl_proc_list_create()** zum Erstellen der Liste, wobei die Liste zu Beginn einen Prozess mit PID = 1 und PPID = 0 enthalten soll,
- die Funktion **int dl_proc_list_insert(dl_proc_list *list, int pid)** zum Einfügen eines Prozesses mit PID **pid** in die Liste, unter den folgenden Bedingungen:
  - Die Liste ist stets **aufsteigend** sortiert,
  - jede Prozess-ID kommt **höchstens** einmal vor und
  - als neuer Elternprozess wird der Prozess mit der **bisher** höchsten Prozess-ID festgelegt.
- die Funktion **int dl_proc_list_get(dl_proc_list *list, int position, int* pid, int* ppid)** zum Abfragen der PID und PPID des Prozesses an der gegebenen Position in der Liste und Abspeichern der erhaltenen Werte in den Parametern **pid** und **ppid**,
- die Funktion **int dl_proc_list_remove(dl_proc_list *list, int pid)** zum Entfernen des Prozesses **p** mit der PID **pid** aus der Liste, unter den folgenden Bedingungen:

  - Das Löschen des Prozesses mit **PID = 1** soll verhindert werden und
  - die Kindprozesse des zu löschenden Prozesses **p** erhalten den Elternprozess von **p** als neuen Elternprozess.
- und die Funktion **void dl_proc_list_free(dl_proc_list *list)** zum Freigeben des allokierten Speichers.

Beachten Sie dabei:

- Geben Sie für die Funktionen der doppelt verketteten Liste im Fehlerfall negative Werte und im Erfolgsfall 0 zurück. Falls der Rückgabetyp ein Pointer ist, soll die Funktion im Fehlerfall **NULL** zurückgeben.

Implementieren Sie anschließend ein Programm `proctable.c` mit der nachfolgenden Funktionalität:

Die Prozesstabelle wird wie folgt **nur** durch die Kommandozeilenargumente des Programms manipuliert:

- Die Kommandozeilenargumente sind als **Prozess-IDs und somit als Integer-Werte** zu interpretieren. Verwenden Sie die Funktion **strtol** zur Umwandlung eines Strings in einen Integer-Wert. Argumente mit dem **Integer-Wert 0** werden ignoriert, geben Sie dabei eine entsprechende Warnung aus.
- **Prozess einfügen:** Ein **positiver** Integer-Wert stellt einen neuen Prozess dar, der in die Liste eingefügt wird. Das Einfügen von vorhandenen Prozess-IDs ist **nicht zugelassen** und soll mit einer Fehlermeldung quittiert werden, ohne dass das Programm abbricht.
- **Prozess löschen:** Ein **negativer** Integer-Wert löscht den Prozess mit dem entsprechenden positiven Wert der ID aus der Liste. Ist der zu löschende Prozess **nicht vorhanden**, soll eine Fehlermeldung ausgegeben werden.
- **Am Ende** des Programms soll der Inhalt der Prozesstabelle, also sowohl die Prozess-ID als auch die ID des Elternprozesses, zeilenweise ausgegeben werden (**siehe unten**).

Beachten Sie bei der **gesamten Aufgabe**:

- Fehler von **Bibliotheksfunktionen** müssen abgefangen werden. Geben Sie eine entsprechende Fehlermeldung aus (vgl. **perror(3)**).
- Alle Fehler- und Warnungsmeldungen müssen auf dem Fehlerausgabekanal (**stderr**) ausgegeben werden.
- Geben Sie **jedweden** allokierten Speicher zum Programmende wieder frei.

Beispielhafter Programmaufruf und Ausgabe:

```
> ./proctable 7 7 -1 3 -8 abc 8
Fehler beim Einfuegen: PID 7 existiert bereits.
Fehler: Prozess mit ID = 1 kann nicht geloescht werden.
Fehler beim Loeschen: Prozess mit ID = 8 existiert nicht.
Warnung: Ungueltiges Argument "abc" ignoriert.
PID = 1, Parent PID = 0
PID = 3, Parent PID = 7
PID = 7, Parent PID = 1
PID = 8, Parent PID = 7
```

(7 Punkte)