Übung 1

Organisatorisches & Prozesse

Max Schrötter
schroetter@cs.uni-potsdam.de

Institute for Computational Science
University of Potsdam

24.10.2025

Agenda

1. Organisatorisches

2. Prozesse & Prozesserzeugung

Wer sind wir?

Übungen:

- Max Schrötter
  - Haus 70, Zimmer 2.10, Sprechzeit nach Vereinbarung
  - E-Mail: maxschro@cs.uni-potsdam.de

- Philipp Ungrund, Andreas Niemann (Tutoren)
  - korrigieren Übungsblätter
  - Tutoren für Hausaufgabenpräsentationen

Übungsbetrieb

Alle Materialien für Übung und Vorlesung unter:

`https://www.cs.uni-potsdam.de/bs/teaching/docs/courses/ws2025/gbr/`

- Üblicher Ablauf:
  - Bei Problemen/Nachfragen Besprechung der Lösungen zum letzten Aufgabenblatt
  - Ergänzungen/Vertiefung/Klärung von Fragen zur Vorlesung
  - Hinweise/Vorbereitung für das nächste Aufgabenblatt

Übungsbetrieb

Alle Materialien für Übung und Vorlesung unter:

https://www.cs.uni-potsdam.de/bs/teaching/docs/courses/ws2025/gbr/

- Üblicher Ablauf:
  - Bei Problemen/Nachfragen Besprechung der Lösungen zum letzten Aufgabenblatt
  - Ergänzungen/Vertiefung/Klärung von Fragen zur Vorlesung
  - Hinweise/Vorbereitung für das nächste Aufgabenblatt
- Folien im Netz, ohne Lösung zu Übungsaufgaben

Übungsbetrieb

Alle Materialien für Übung und Vorlesung unter:

https://www.cs.uni-potsdam.de/bs/teaching/docs/courses/ws2025/gbr/

- Üblicher Ablauf:
  - Bei Problemen/Nachfragen Besprechung der Lösungen zum letzten Aufgabenblatt
  - Ergänzungen/Vertiefung/Klärung von Fragen zur Vorlesung
  - Hinweise/Vorbereitung für das nächste Aufgabenblatt
- Folien im Netz, ohne Lösung zu Übungsaufgaben
- voraussichtlich (!) 6 Übungsblätter
  voraussichtlich (!) jeweils 20 Punkte

Übungsbetrieb

Alle Materialien für Übung und Vorlesung unter:

https://www.cs.uni-potsdam.de/bs/teaching/docs/courses/ws2025/gbr/

- Üblicher Ablauf:
  - Bei Problemen/Nachfragen Besprechung der Lösungen zum letzten Aufgabenblatt
  - Ergänzungen/Vertiefung/Klärung von Fragen zur Vorlesung
  - Hinweise/Vorbereitung für das nächste Aufgabenblatt
- Folien im Netz, ohne Lösung zu Übungsaufgaben
- voraussichtlich (!) 6 Übungsblätter
  voraussichtlich (!) jeweils 20 Punkte
- Verwaltung der Übungspunkte in Moodle

Abgabe der Übungen

- Gruppenarbeit: 2-3 Personen pro Gruppe;
  Einzelabgaben nicht gestattet → ggf. Zwangszusammenlegung;
  Gruppen nur nach Absprache änderbar

Abgabe der Übungen

- Gruppenarbeit: 2-3 Personen pro Gruppe;
  Einzelabgaben nicht gestattet → ggf. Zwangszusammenlegung;
  Gruppen nur nach Absprache änderbar
- Abgaben der Hausaufgaben:
  1. Theorieaufgaben: als PDF-Datei via Moodle
  2. Praxisaufgaben via Git.UP, siehe erstes Tutorium
     Bei mehreren Dateien: (Dateiname = Blatt-Nr.+Gruppen-Nr., z.B.
     `blatt1_gruppe5.tar.gz` mit Unterverzeichnissen für die einzelnen Aufgaben)
     `https://moodle2.uni-potsdam.de/course/view.php?id=47479`

Abgabe der Übungen

- Gruppenarbeit: 2-3 Personen pro Gruppe;
  Einzelabgaben nicht gestattet → ggf. Zwangszusammenlegung;
  Gruppen nur nach Absprache änderbar
- Abgaben der Hausaufgaben:
  1. Theorieaufgaben: als PDF-Datei via Moodle
  2. Praxisaufgaben via Git.UP, siehe erstes Tutorium
     Bei mehreren Dateien: (Dateiname = Blatt-Nr.+Gruppen-Nr., z.B.
     `blatt1_gruppe5.tar.gz` mit Unterverzeichnissen für die einzelnen Aufgaben)
     `https://moodle2.uni-potsdam.de/course/view.php?id=47479`
- Bei Praxisaufgaben wichtig:
  - Immer Makefile mit abgeben, das ausführbare Dateien erstellt
    Zu Makefiles siehe auch Vorlesung C- und Unix-Tools, Kapitel 4
  - Programme müssen auf `tiree.lab.cs.uni-potsdam.de` kompilieren (SSH-Zugang)
    Hinweis: Der Zugang für alle zur Lehrveranstaltung angemeldeten Studenten ist über den Instituts-Account freigeschaltet.

Was sind die Anforderungen?

Allgemein:

- Es müssen studienbegleitend mindestens 50% der Hausaufgabenpunkte erreicht werden
- Für 3 von 6 eingereichten Hausaufgaben muss mindestens eine Aufgabe erfolgreich verteidigt werden.

Konkret für die Hausaufgaben:

- Geben Sie lesbare Antworten ab!
- selbstgeschriebener Code, selbstverfasste Antworten (keine gruppenübergreifenden Abgaben)
- kein Kopieren von ChatGPT, Hintergründe müssen verstanden werden!

Was sind die Anforderungen?

Konkret für die Präsentationen:

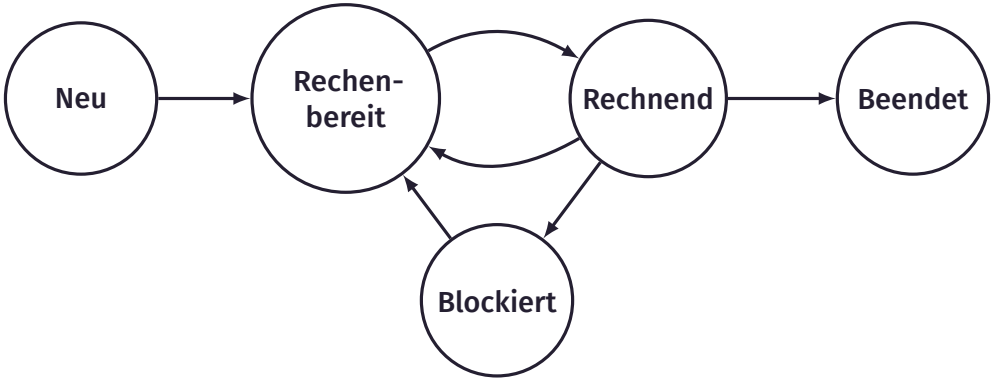
- In den geraden Wochen (ab Woche 4) müssen sich alle Gruppen auf einen Termin zur Präsentation ihrer Lösungen via Moodle anmelden. (mindestens 72h vorher)
- Die Präsentationen (15 min per Gruppe) finden Freitags von 14-16 Uhr statt. (falls nötig werden weitere Termine angeboten)
- Jede Studierende jeder Gruppe muss eine zufällige Aufgabe des Aufgabenblattes präsentieren/erklären können.

Agenda

1. Organisatorisches

2. Prozesse & Prozesserzeugung

Prozesszustände



Prozesszustände

Beispielaufgabe:

- Das System hat 1 Festplatte und nutzt eine Warteschlange für Festplattenzugriffe (FCFS)
- Latenz für Festplattenzugriff: 30ms Start: 2 Prozesse A und B mit einer Laufzeit von jeweils 20ms
- Nachdem Prozess A 10ms lief, greift er auf die Festplatte zu
- Nachdem Prozess B 15ms lief, greift er auf die Festplatte zu

Hinweis: Falls 2 Prozesse simultan eintreffen, werden diese in die Warteschlange alphabetisch aufsteigend einsortiert.

Prozesszustände - Beispielaufgabe

A: Running (Active), R: Ready, B: Blocked, C: Completed

Zeit (ms)	Prozesse A	Prozesse B	Ready-Queue	HDD-Queue	Notizen

Prozesszustände - Beispielaufgabe

A: Running (Active), R: Ready, B: Blocked, C: Completed

Zeit (ms)	Prozesse A	Prozesse B	Ready-Queue	HDD-Queue	Notizen
0	A	R	B		

Prozesszustände - Beispielaufgabe

A: Running (Active), R: Ready, B: Blocked, C: Completed

Zeit (ms)	Prozesse A	Prozesse B	Ready-Queue	HDD-Queue	Notizen
0	A	R	B		
10	B	A		A	

Prozesszustände - Beispielaufgabe

A: Running (Active), R: Ready, B: Blocked, C: Completed

Zeit (ms)	Prozesse A	Prozesse B	Ready-Queue	HDD-Queue	Notizen
0	A	R	B		
10	B	A		A	
25	B	B		A,B	

Prozesszustände - Beispielaufgabe

A: Running (Active), R: Ready, B: Blocked, C: Completed

Zeit (ms)	Prozesse A	Prozesse B	Ready-Queue	HDD-Queue	Notizen
0	A	R	B		
10	B	A		A	
25	B	B		A,B	
40	A	B		B	A finishes HDD

Prozesszustände - Beispielaufgabe

A: Running (Active), R: Ready, B: Blocked, C: Completed

Zeit (ms)	Prozesse A	Prozesse B	Ready-Queue	HDD-Queue	Notizen
0	A	R	B		
10	B	A		A	
25	B	B		A,B	
40	A	B		B	A finishes HDD
50	C	B		B	

Prozesszustände - Beispielaufgabe

A: Running (Active), R: Ready, B: Blocked, C: Completed

Zeit (ms)	Prozesse A	Prozesse B	Ready-Queue	HDD-Queue	Notizen
0	A	R	B		
10	B	A		A	
25	B	B		A,B	
40	A	B		B	A finishes HDD
50	C	B		B	
70	C	A			B finishes HDD

Prozesszustände - Beispielaufgabe

A: Running (Active), R: Ready, B: Blocked, C: Completed

Zeit (ms)	Prozesse A	Prozesse B	Ready-Queue	HDD-Queue	Notizen
0	A	R	B		
10	B	A		A	
25	B	B		A,B	
40	A	B		B	A finishes HDD
50	C	B		B	
70	C	A			B finishes HDD
75	C	C			

PID

- **pid**: Identifikator eines Prozesses

```
pid_t pid = getpid();
```

- **ppid**: Identifikator des Elternprozesses

```
pid_t ppid = getppid();
```

PID

- **pid**: Identifikator eines Prozesses

```
pid_t pid = getpid();
```

- **ppid**: Identifikator des Elternprozesses

```
pid_t ppid = getppid();
```

- es existieren **nur Syscalls** für die PID und PPID

Prozessbaum

```
$ps -ef --forest
UID          PID    PPID  C STIME TTY          TIME CMD
mschroe+  144583       1  0 14:35 ?        00:00:00 /usr/bin/zsh
mschroe+  144593  144583  0 14:35 ?        00:00:00  \_ xterm
mschroe+  144595  144593  1 14:35 pts/7    00:00:03      \_ zsh
mschroe+  145281  144595  9 14:37 pts/7    00:00:06          \_ /usr/lib/chromium/chromium google.com
mschroe+  145290  145281  0 14:37 pts/7    00:00:00          |   \_ /usr/lib/chromium/chromium
mschroe+  145315  145290  1 14:37 pts/7    00:00:01          |   |   \_ /usr/lib/chromium/chromium
mschroe+  145291  145281  0 14:37 pts/7    00:00:00          |   \_ /usr/lib/chromium/chromium
mschroe+  145293  145291  0 14:37 pts/7    00:00:00          |   |   \_ /usr/lib/chromium/chromium
mschroe+  145321  145293  0 14:37 pts/7    00:00:00          |   |       \_ /usr/lib/chromium/chromium
mschroe+  145346  145293  4 14:37 pts/7    00:00:02          |   |       \_ /usr/lib/chromium/chromium
mschroe+  145318  145281  1 14:37 pts/7    00:00:00          |   \_ /usr/lib/chromium/chromium
mschroe+  145665  145281  0 14:37 pts/7    00:00:00          |   \_ /usr/lib/chromium/chromium
mschroe+  146017  144595 10 14:37 pts/7    00:00:05          \_ /usr/lib/firefox/firefox /tmp/callbacks.html
mschroe+  146103  146017  0 14:37 pts/7    00:00:00          |   \_ /usr/lib/firefox/firefox
mschroe+  146122  146017  0 14:37 pts/7    00:00:00          |   \_ /usr/lib/firefox/firefox
mschroe+  146271  146017  0 14:37 pts/7    00:00:00          |   \_ /usr/lib/firefox/firefox
mschroe+  146317  144595  0 14:38 pts/7    00:00:00          \_ nvim test.c
mschroe+  146320  146317  4 14:38 ?        00:00:01              \_ nvim --embed test.c
mschroe+  146335  146320  8 14:38 ?        00:00:02                  \_ node
↪  /home/mschroetter/.local/share/nvim/lazy/copilot.lua/copilot/index.js$
```

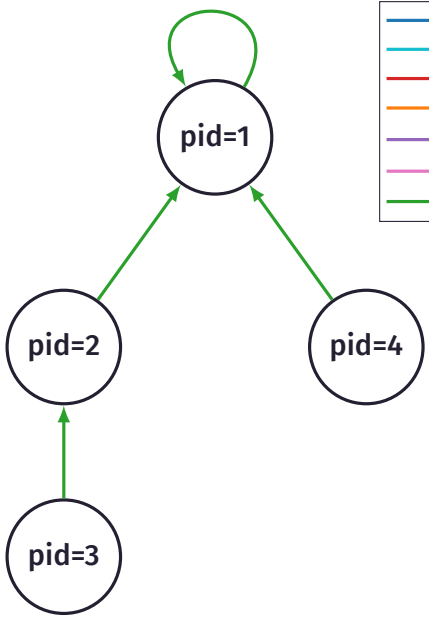
Prozessbaum

```
$ps -ef --forest
UID          PID    PPID  C STIME TTY          TIME CMD
mschroe+  144583       1  0 14:35 ?        00:00:00 /usr/bin/zsh
mschroe+  144593  144583  0 14:35 ?        00:00:00  \_ xterm
mschroe+  144595  144593  1 14:35 pts/7    00:00:03      \_ zsh
mschroe+  145281  144595  9 14:37 pts/7    00:00:06          \_ /usr/lib/chromium/chromium google.com
mschroe+  145290  145281  0 14:37 pts/7    00:00:00          |   \_ /usr/lib/chromium/chromium
mschroe+  145315  145290  1 14:37 pts/7    00:00:01          |   |   \_ /usr/lib/chromium/chromium
mschroe+  145291  145281  0 14:37 pts/7    00:00:00          |   \_ /usr/lib/chromium/chromium
mschroe+  145293  145291  0 14:37 pts/7    00:00:00          |   |   \_ /usr/lib/chromium/chromium
mschroe+  145321  145293  0 14:37 pts/7    00:00:00          |   |       \_ /usr/lib/chromium/chromium
mschroe+  145346  145293  4 14:37 pts/7    00:00:02          |   |       \_ /usr/lib/chromium/chromium
mschroe+  145318  145281  1 14:37 pts/7    00:00:00          |   \_ /usr/lib/chromium/chromium
mschroe+  145665  145281  0 14:37 pts/7    00:00:00          |   \_ /usr/lib/chromium/chromium
mschroe+  146017  144595 10 14:37 pts/7    00:00:05          \_ /usr/lib/firefox/firefox /tmp/callbacks.html
mschroe+  146103  146017  0 14:37 pts/7    00:00:00          |   \_ /usr/lib/firefox/firefox
mschroe+  146122  146017  0 14:37 pts/7    00:00:00          |   \_ /usr/lib/firefox/firefox
mschroe+  146271  146017  0 14:37 pts/7    00:00:00          |   \_ /usr/lib/firefox/firefox
mschroe+  146317  144595  0 14:38 pts/7    00:00:00          \_ nvim test.c
mschroe+  146320  146317  4 14:38 ?        00:00:01              \_ nvim --embed test.c
mschroe+  146335  146320  8 14:38 ?        00:00:02                  \_ node
↪  /home/mschroetter/.local/share/nvim/lazy/copilot.lua/copilot/index.js$
```

→Prozessbeziehungen sind **“tree-like”**

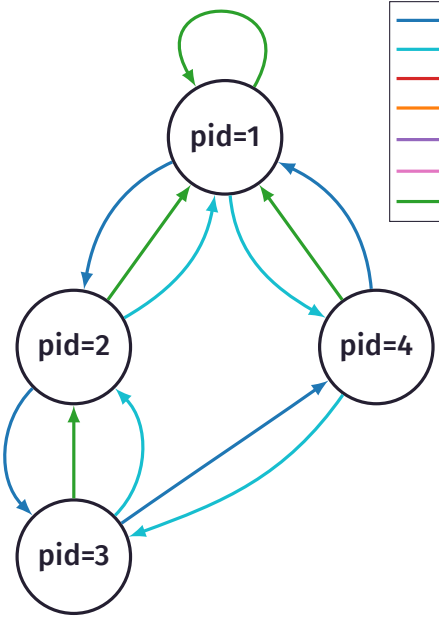
Linux Prozesstabelle

```
struct task_struct {
   ...
  struct list_head tasks;
  pid_t pid;
  struct task_struct *parent;
  struct list_head children;
  struct list_head sibling;
   ...
}

struct list_head {
  struct list_head *next, *prev;
};
```



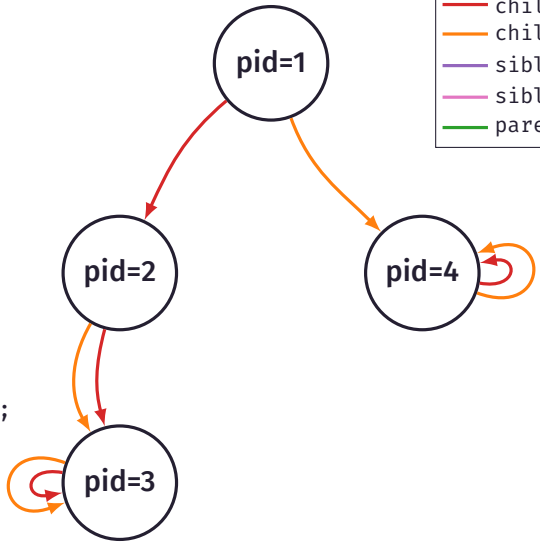
Linux Prozesstabelle

```
struct task_struct {
   ...
  struct list_head tasks;
  pid_t pid;
  struct task_struct *parent;
  struct list_head children;
  struct list_head sibling;
   ...
}

struct list_head {
  struct list_head *next, *prev;
};
```



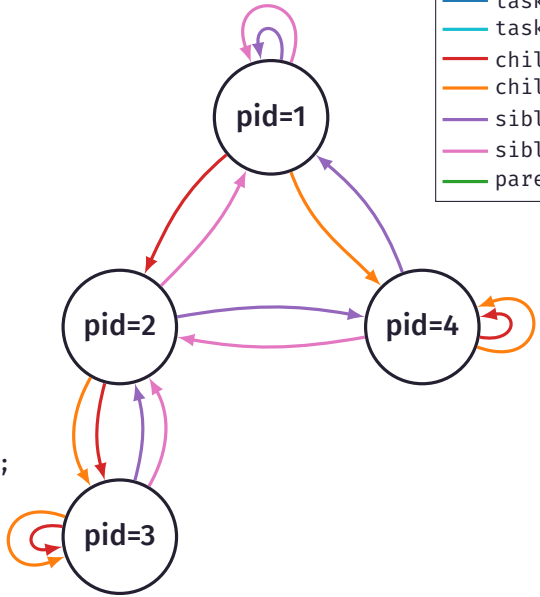
Linux Prozesstabelle

```
struct task_struct {
   ...
  struct list_head tasks;
  pid_t pid;
  struct task_struct *parent;
  struct list_head children;
  struct list_head sibling;
   ...
}

struct list_head {
  struct list_head *next, *prev;
};
```



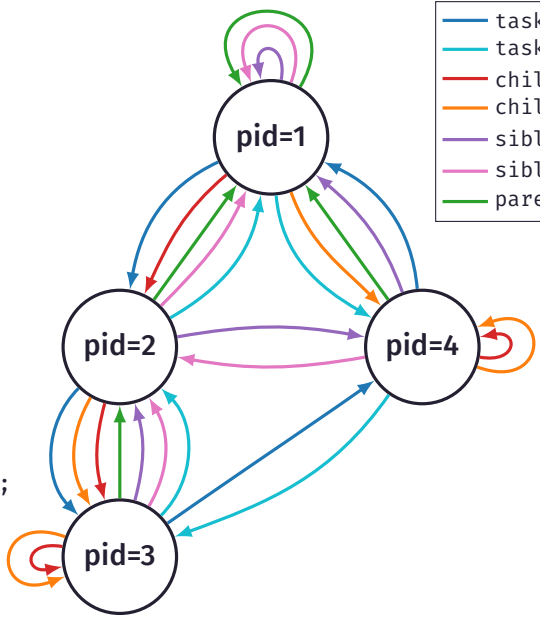
Linux Prozesstabelle

```
struct task_struct {
   ...
  struct list_head tasks;
  pid_t pid;
  struct task_struct *parent;
  struct list_head children;
  struct list_head sibling;
   ...
}

struct list_head {
  struct list_head *next, *prev;
};
```



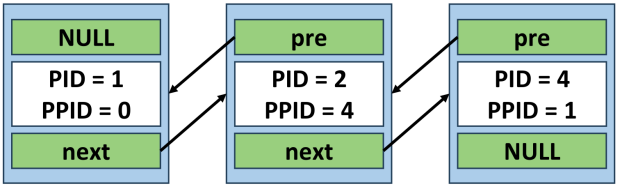
Linux Prozesstabelle

```
struct task_struct {
   ...
  struct list_head tasks;
  pid_t pid;
  struct task_struct *parent;
  struct list_head children;
  struct list_head sibling;
   ...
}

struct list_head {
  struct list_head *next, *prev;
};
```



Aufgabe Prozesstabelle

- Implementierung einer vereinfachten Prozesstabelle in C
- Ziel: Wiederholung C-Programmierung & Verstehen von bekannten Kernel Datenstrukturen
- Datenstruktur: ist euch überlassen (**mindestens doppelt verkettete Liste**)
- Eigenschaften wie Kind-Beziehung müssen nicht im Datentyp abgebildet werden, können auch über Funktionen realisiert werden (itterieren über alle Prozesse etc.)



Literaturempfehlung

- *The Linux Programming Interface*, Michael Kerrisk, Kapitels 6, **24, 26**

- man pages:
  - `man 1 ps`

Noch Fragen?

Max Schrötter

Potsdam, 24.10.2025

schroetter@cs.uni-potsdam.de