

Kapitel 2: Der Prozess

2: Der Prozess

Der Prozess

Programm während der Ausführung.

- ▶ Wird ein Programm aufgerufen, dann wird es in den Hauptspeicher geladen und gestartet.
- ▶ Das ablaufende Programm wird als **Prozess** bezeichnet.
- ▶ Ein Programm kann mehrmals gleichzeitig gestartet werden. (Beispiel: Benutzer öffnet mehrere Terminals)
- ▶ **Resultat: Mehrere verschiedene Prozesse.**

```
$ ps
21876 pts/1      00:00:01 bash
26451 pts/1      00:00:06 emacs
30387 pts/1      00:00:05 emacs
30587 pts/1      00:00:00 ps
```

Der Prozesskontext

Bei Prozessen gibt es drei Arten von Kontextinformationen:

1. **Benutzerkontext**: Daten des Prozesses im zugewiesenen Adressraum (virtueller Speicher)
2. **Hardwarekontext**: CPU-Register und CPU-Cache
3. **Systemkontext**: Metadaten die das Betriebssystem über einen Prozess speichert
 - ▶ Prozess-ID (PID)
 - ▶ Laufzeit
 - ▶ Zugriffsrechte
 - ▶ Prioritäten
 - ▶ Geöffnete Dateien
 - ▶ Zugeordnete Geräte
 - ▶ ...

Prozesse

- ▶ Jeder Prozess hat seinen eigenen Prozesskontext, der von den Kontexten der anderen Prozesse meist unabhängig ist.
- ▶ Gibt ein Prozess den Prozessor ab, wird sein Kontext, also der Inhalt der CPU-Register, zwischengespeichert.
- ▶ Erhält der Prozess wieder den Zugriff auf die CPU, wird der Inhalt des Kontext wiederhergestellt und die Register werden mit den zuvor gespeicherten Daten geladen.
- ▶ Prozessmanagement und Prozessinteraktion ermöglichen (präemptives) Multitasking.

Prozesserzeugung

- ▶ Ein Computer-System muss in der Lage sein Prozesse zu erzeugen.
- ▶ Ereignisse die einen Prozess erzeugen:
 - ▶ Initialisierung des Systems
 - ▶ (Eltern-)Prozesse können (Kind-)Prozesse starten.
 - ▶ Bei interaktiven Systemen können Anwender Prozesse starten.
 - ▶ Initiierung einer Stapelverarbeitung (Großrechner).
- ▶ Prozesse werden mittels Systemcalls erzeugt.
- ▶ Es wird zwischen Vordergrund- und Hintergrundprozessen unterschieden.
- ▶ Hintergrundprozesse werden auch **Daemons** oder **Services** genannt.

Prozessbeendigung

Prozesse terminieren aufgrund der folgenden Bedingungen:

- ▶ Normales Beenden (freiwillig)

Aufrufen der `exit()`-Methode

- ▶ Beenden aufgrund eines Fehlers (freiwillig)

```
rm -f foo.c; gcc -c foo.c
```

- ▶ Beenden aufgrund eines schwerwiegenden Fehlers (unfreiwillig)

```
dst=NULL; memcpy(dst, src, 9001);
```

- ▶ Beenden durch einen anderen Prozess (unfreiwillig)

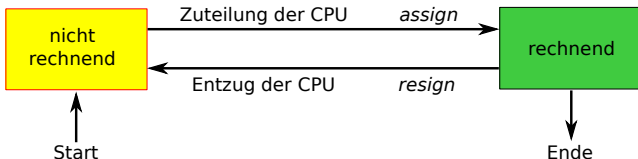
```
killall foo
```

2.1: Prozesszustände

- ▶ Ein Prozess wird zu Beginn der Programmausführung erzeugt und bei der Terminierung des Programms beendet.
- ▶ Die Ausführung eines Prozesses kann zur Erzeugung bzw. Beendigung weiterer Prozesse führen.
- ▶ Jeder Prozess befindet sich zu jedem Zeitpunkt in einem Zustand.
- ▶ Wie viele unterschiedliche Zustände es gibt, hängt vom Prozessmodell des Betriebssystems ab.

Frage: Wie viele Zustände benötigt ein minimales Prozessmodell?

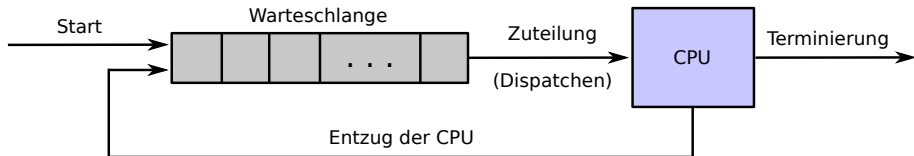
Das 2-Zustands-Prozessmodell



- ▶ Prinzipiell genügen zwei Prozesszustände
 - ▶ **Rechnend**: Einem Prozess wurde die CPU zugeteilt.
 - ▶ **Nicht rechnend**: Der Prozess wartet auf die Zuteilung der CPU.
- ▶ **Kontextwechsel**: Das Betriebssystem tauscht den **rechnend** Prozess durch einen **nicht rechnend** Prozess aus.

Frage: Wie lässt sich ein Kontextwechsel umsetzen?

Das 2-Zustands-Prozessmodell: Implementierung



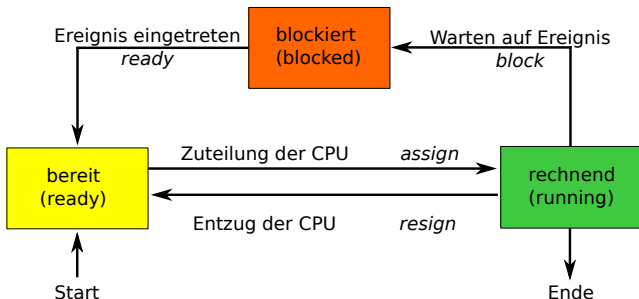
- ▶ Prozesse im Zustand **nicht rechnend** werden in einer Warteschlange *geparkt*.
- ▶ Der **Scheduler** bestimmt die Reihenfolge, in der die Prozesse Zugriff auf die CPU erhalten.
- ▶ Der Scheduler implementiert ein **Scheduling-Verfahren**.
- ▶ Der **Dispatcher** setzt die Zustandsübergänge (Kontextwechsel) um.

Das 2-Zustands-Prozessmodell: Diskussion

- ▶ Das 2-Zustands-Prozessmodell geht davon aus, dass alle Prozesse immer zur Ausführung bereit sind.
- ▶ Dies ist aber nur bei sehr wenigen System der Fall.
- ▶ In der Regel können Prozesse **blockieren**.
 - ▶ Warten auf I/O (Benutzereingabe, Netzwerkpakete, ...)
 - ▶ Ressource ist von einem anderen Prozess belegt (Speicher, Drucker, ...)
- ▶ Der Zustand **nicht rechnend** wird daher in zwei Zustände aufgeteilt.
 - ▶ **bereit (ready)**
 - ▶ **blockiert (blocked)**

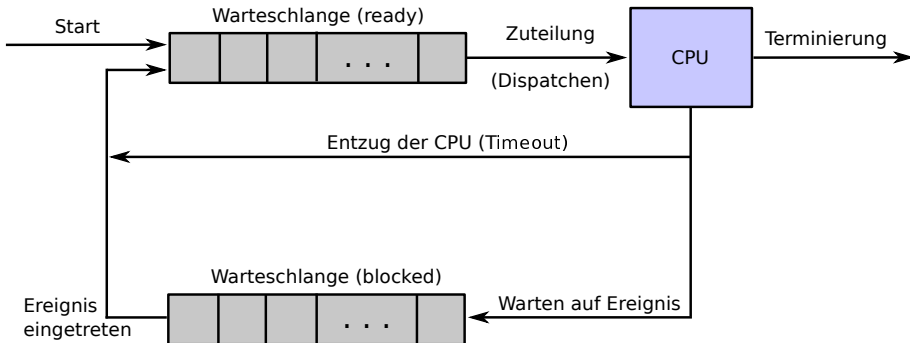
⇒ 3-Zustands-Prozessmodell

3-Zustands-Prozessmodell

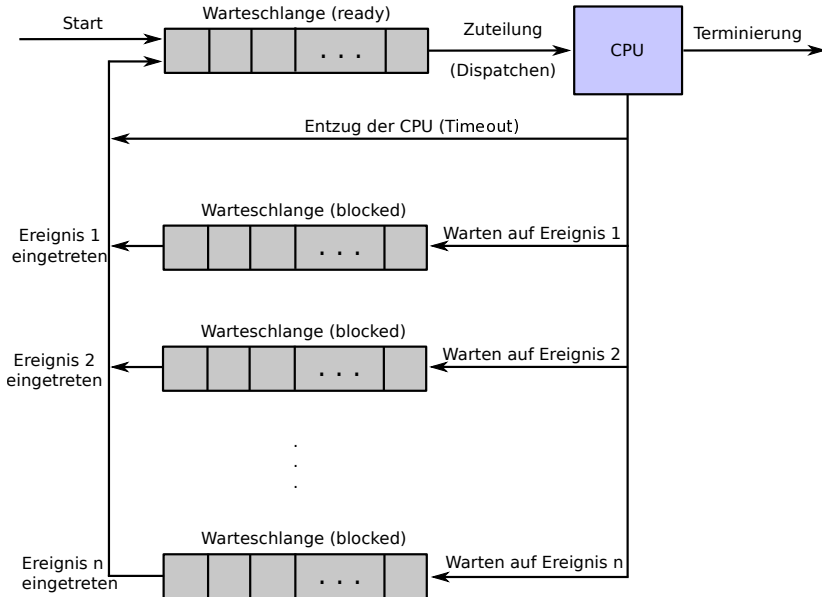


- ▶ **Block:** Der rechnende Prozess wartet auf eine Nachricht oder ein Ereignis und wechselt in den Zustand blockiert.
- ▶ **Assign:** Der Scheduler teilt dem Prozess die CPU zu.
- ▶ **Resign:** Der Scheduler entzieht dem Prozess die CPU.
- ▶ **Ready:** Es tritt ein Ereignis ein das den Prozess reaktiviert.

3-Zustands-Prozessmodell: Implementierung (1/2)



3-Zustands-Prozessmodell: Implementierung (2/2)

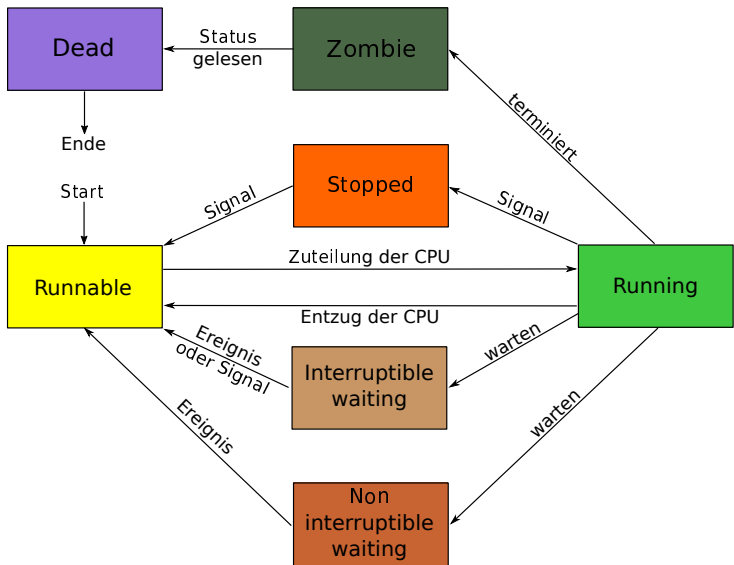


Prozesszustände unter Linux

- ▶ **D**: Uninterruptible sleep (usually IO)
- ▶ **R**: Running or runnable (on run queue)
- ▶ **S**: Interruptible sleep (waiting for an event to complete)
- ▶ **T**: Stopped by job control signal
- ▶ **X**: Dead (should never be seen)
- ▶ **Z**: Defunct (“zombie”) process, terminated but not killed by its parent

Quelle: Manpage von ps

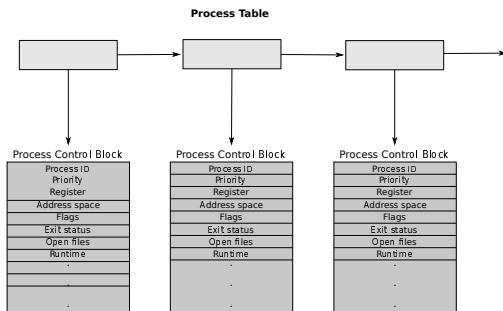
Linux Prozessmodell



Prozessverwaltung

- ▶ Das Betriebssystem verwaltet Prozesse
- ▶ Dazu muss es quasi alles über den Prozess wissen
 - ▶ Prozesszustand (D, R, S, T, X oder Z)
 - ▶ Reservierte Speicherbereiche
 - ▶ Geöffnete Dateien und Netzwerkverbindungen
 - ▶ ...
- ▶ Wird ein Prozess angehalten, muss er zu einem späteren Zeitpunkt in dem Zustand, in dem er gestoppt wurde, wieder aktiviert werden.
- ▶ Daher müssen der Zustand eines Prozesses für die Dauer der Unterbrechung zwischengespeichert werden.

Prozesskontrollblock (1/2)



- ▶ **Prozesstabelle**: Liste im Kernel die zu jedem Prozess einen Zeiger auf dessen **Prozesskontrollblock** hat.
- ▶ **Prozesskontrollblock (PCB)**: Enthält den kompletten Zustand eines Prozesses mit Ausnahme des Inhalts des Adressraums.

Prozesskontrollblock (2/2)

Im Prozesskontrollblock jedes Prozesses befinden sich die folgenden Kontextinformationen:

- ▶ Die eindeutige Prozessidentifikation (PID)
- ▶ Der aktuelle Prozesszustand
- ▶ Inhalt der Prozessorregister (Befehlszähler, Stack Pointer, usw.)
- ▶ Eine Liste mit zugeordneten Bereichen im Hauptspeicher
- ▶ Vom Prozess geöffnete Daten und zugeordnete Peripheriegeräte
- ▶ Verwaltungs-/Schedulinginformationen (Priorität, Laufzeit, usw.)

Zustandsübergänge

- ▶ Für einen Zustandsübergang eines Prozesses wird der PCB des betreffenden Prozesses aus der alten Zustandsliste entfernt und in die neue Zustandsliste eingefügt.
- ▶ Für Prozesse im Zustand **rechnend** existiert keine eigene Liste, aber ein PCB-Eintrag in der Prozesstabelle.
- ▶ Beim Übergang eines Prozesses in den Zustand **rechnend** wird der Kontext des Prozessors aus dem PCB geladen.
- ▶ Beim Übergang aus dem Zustand **rechnend** in einen anderen Zustand werden die Registerinhalte des Prozessors in den PCB gesichert (gerettet).

Zusammenfassung

Nach diesem Kapitel sollten Sie ...

- ▶ ... verstanden haben was ein Prozess ist.
- ▶ ... das 3-Zustands-Prozessmodell kennen.
- ▶ ... das Linux-Prozessmodell kennen.
- ▶ ... wissen was ein Prozesskontrollblock ist.
- ▶ ... verstanden haben wie unter Linux der Zustandsübergang eines Prozesses funktioniert.