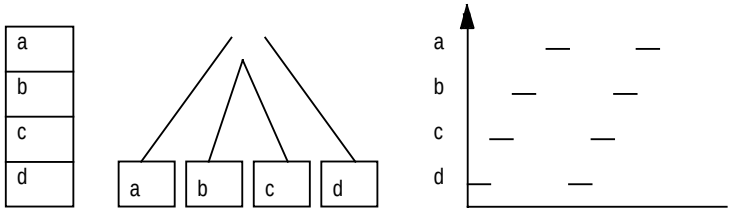
2. Betriebssystem-Kern

2.1. Prozesse

- Prozeß ist einziger aktiver Objekttyp eines Betriebssystems
- enthält IP, Register, Variablen, Stack, Code (Programm), E/A und *Zustände*
= Programm während der Ausführung
- jeder Prozeß hat seine eigene virtuelle CPU; reale CPU wird zwischen den einzelnen Prozessen geteilt.
- quasiparallel
- Aus Sicht des Anwenders parallel

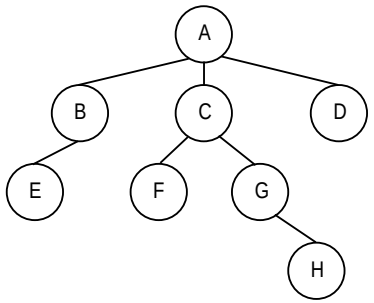


- Zuteilungsstrategien (Schedulingstrategien) entscheiden, wann Umschaltung zwischen Prozessen erfolgt.

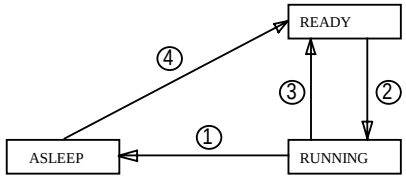
2.2. Prozeßhierachie

- ∃ Mechanismus zur Erzeugung von Prozessen

- UNIX: fork kreiert neuen Sohnprozeß,
  Sohn kann weitere Prozesse kreieren
  ⇒ Baum von Prozessen entsteht
- Jeder Prozeß hat einen Elternprozeß
- Jeder Sohn hat nur einen Vaterprozeß
- Jeder Prozeß kann weitere Prozesse kreieren ⇒ child process
- Alle im System befindlichen Prozesse bilden Baumstruktur



2.3. Allgemeines Prozeßzustandsmodell



Ganz simpel:

Running: im Augenblick wird CPU benutzt

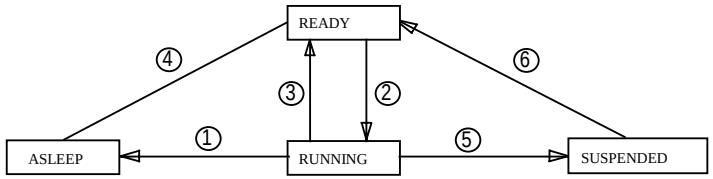
Ready: lauffähig, zeitweise gestoppt während andere Prozesse laufen

Asleep: wartet auf externes Ereignis (Aussage ist sehr verbal bzw. abstrakt)

Übergänge:

(1) Prozesse blockieren, wenn sie z.B. auf eine Eingabe warten
(2) Der Scheduler wählt einen anderen Prozeß
(3) Der Scheduler wählt diesen Prozeß
(4) Die Eingabe wird verfügbar

Bei einigen Systemen kommt der Zustand ***suspended*** hinzu (z.B. RMOS)



Jetzt etwas genauer:

- Aus der Sicht der Betriebssysteme kann ein Prozeß sich in einem der folgenden Zustände befinden:

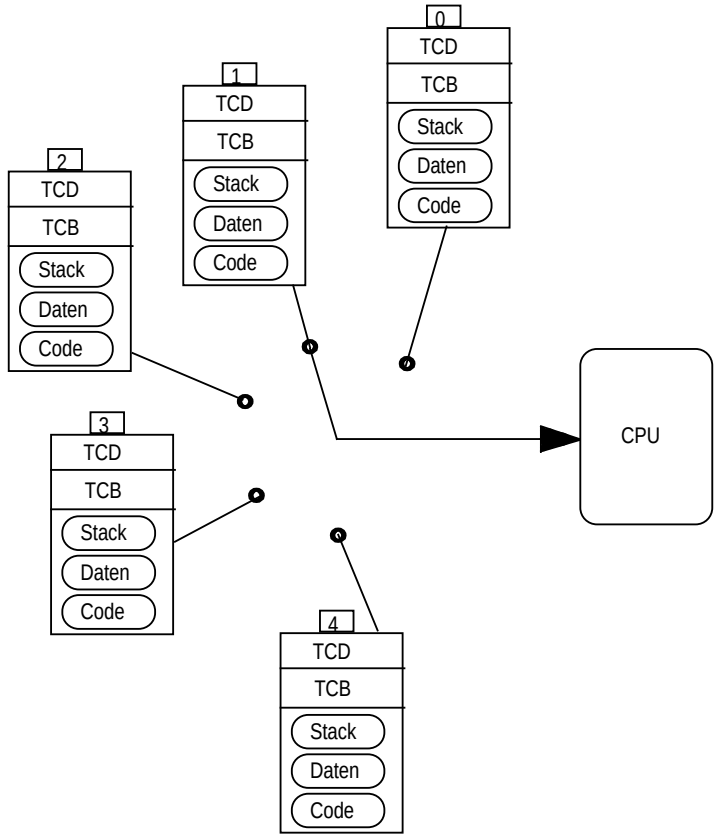
1. running (active, rechnend)	Der Prozeß wird aktuell abgearbeitet; sie "besitzt" die CPU
2. ready (bereit)	Der Prozeß befindet sich ablaufbereiten Zustand. Der Prozeß kann jederzeit vom Prozeß aufgegriffen werden. Sie wird daran gehindert, weil ein höherpriorer Prozeß rechnend ist (s. Prio.bas. Sched.) bzw. weil der Prozeß nicht im Besitz der Zeitscheibe ist (s. Time-Sharing-Sched.)
3. asleep (blocked, sleeping, blockiert, schlafend)	Wartet auf ein nicht verfügbares BM bzw. hat sich selbst in diesen Zustand versetzt.
4. suspended (dormant, suspendiert)	Der Prozeß ist nicht mehr am Ablaufgeschehen beteiligt, ist dem BS aber noch über seine Kontrollstrukturen (Prozeßkontrollblock) bekannt. Eine Aktivierung dieses Prozeß muß explizit verlangt werden-Aktivierung geschieht durch einen anderen Prozeß.

- Einige BS unterscheiden den Zustand Suspendiert noch durch den hervorrufenden Übergang.
- z.B. gibt es in RMX die Zustände asleep-suspended und (ready-)suspended.

2.3 Prozeßkontrollblock

- System legt zu jedem kreierten Prozeß einen Prozeßkontrollblock PCB an
- PCB = Datenstruktur, die Informationen zur Verwaltung von Prozessen enthält
  - momentanen Prozeßzustand
  - eindeutige Prozeßkennung (PID, TOKEN)
  - Priorität des Prozesses
  - Hauptspeicheradresse des Prozesses
  - vom Prozeß reservierte Ressourcen (z.B. Drucker, Dateien)
- BS verwendet die im PCB enthaltene Information um einen Prozeß an einer unterbrochenen Stelle weiterarbeiten zu lassen.
- Entzieht BS einem Prozeß die CPU, so schreibt BS Daten die zum Fortschreiten des Prozesses notwendig sind in den PCB

- Bsp. RMOS
  TCD (Task control block)
  Folie Prozeßumschalter



2.4. Operationen auf Prozesse

1. Prozeß kreieren
2. Prozeß löschen
3. Prozeß blockieren
4. Prozeß deblockieren
5. Prozeß suspendieren
6. Suspendierten Prozeß wiederbeleben
7. Zuteilen der CPU an einen bereiten Prozeß
8. Priorität des Prozesses ändern

Bsp.:

	UNIX	Windows-NT-Prozesse	Windows-NT-Threads	RMOS
1.	fork	CreateProcess	CreateThread	create
2.	exit	ExitProcess	ExitThread	endt
3.	kill	TerminateProcess	TerminateThread	delete
4.	wait			strt (mit Param. Warten auf Ende des Sohnes)
5.	exec			
6.			SuspendThread	suspend
7.			ResumeThread	resume
8.	sleep			pause

Was passiert beim Kreieren eines Prozesses?

- PID
- Eintrag in Prozeßtabelle
- Priorität festlegen
- Anlegen PCB
- Reservierung der benötigten BM

Was passiert beim Löschen eines Prozesses?

- Freigabe der vom Prozeß benötigten BM
- Zerstören PCB
- Löschen aller Kinder, Enkel, etc. (bei UNIX

WindowsNT:

CreateProcess:
Prozeß wird durch den Thread eines anderen Prozesses mittels CreateProcess erzeugt.
Beim Aufruf erhält der Prozeß vom Betriebssystem einen 4GB umfassenden virtuellen Adreßbereich zur Verfügung und lädt diesen dort hinein. Danach erzeugt das System für diesen Prozeß den primären Thread, der mit der Ausführung des Startcodes der C-Laufzeitbibliothek beginnt.
Damit erfolgt der Aufruf der WinMain-Funktion des Programms.
Bei fehlerfreier Ausführung des neuen Prozesses gibt CreateProcess TRUE zurück.

ExitProcess:
Ein Prozeß wird beendet falls einer seiner Threads die Funktion ExitProcess aufruft.

Objekte der WindowsNT- Exekutive

Ereignisobjekte
Pipe-Objekte
Dateiabbildungs-Objekte
Prozeßobjekte
Dateiobjekte
Semaphoren-Objekte
Thread-Objekte
Mutex-Objekte