Informatik
Hauptcampus

H O C H
S C H U L E
T R I E R

Systemadministration
Teil 5

Prof. Dr.-Ing. Jörn Schneider

WIEDERHOLUNG

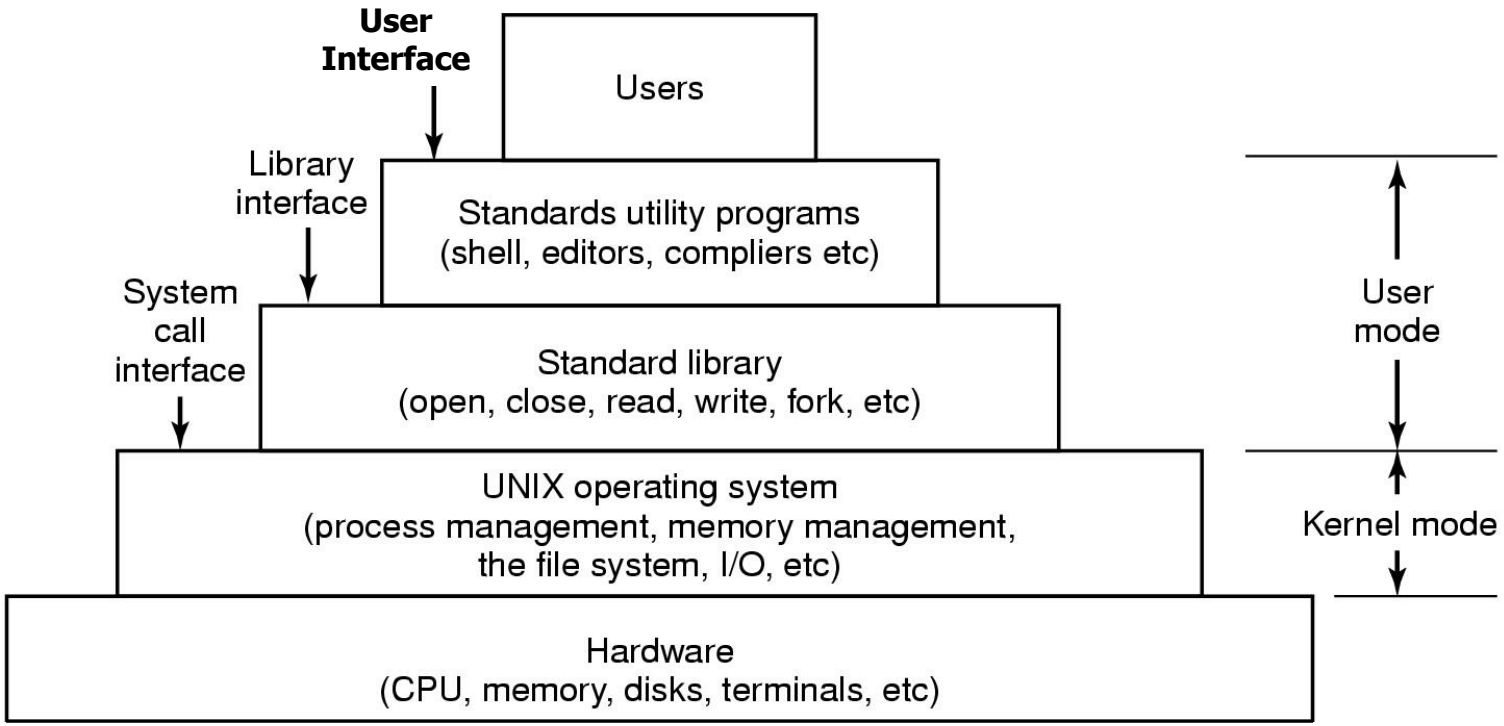
Konzepte zum Thema Benutzer

- Anmeldung
  - Authentifizierung
  - Ausführungsumgebung
- Rechte
  - CPU
  - Speicher
  - Prozesse
  - Dateien
  - Speicherplatz

Rechteebenen

- Hardware
- Betriebssystem
- Systemprogramme
- Anwendungssoftware

UNIX


User Interface
Users
Library interface
Standards utility programs (shell, editors, compliers etc)
System call interface
Standard library (open, close, read, write, fork, etc)
UNIX operating system (process management, memory management, the file system, I/O, etc)
Hardware (CPU, memory, disks, terminals, etc)
User mode
Kernel mode

Rechte - Hardware

- CPU
  - SuperVisor Modus (Alle Register lese- und schreibbar)
  - User Mode
- Speicher
  - Memory Management Unit
- BIOS / Firmware

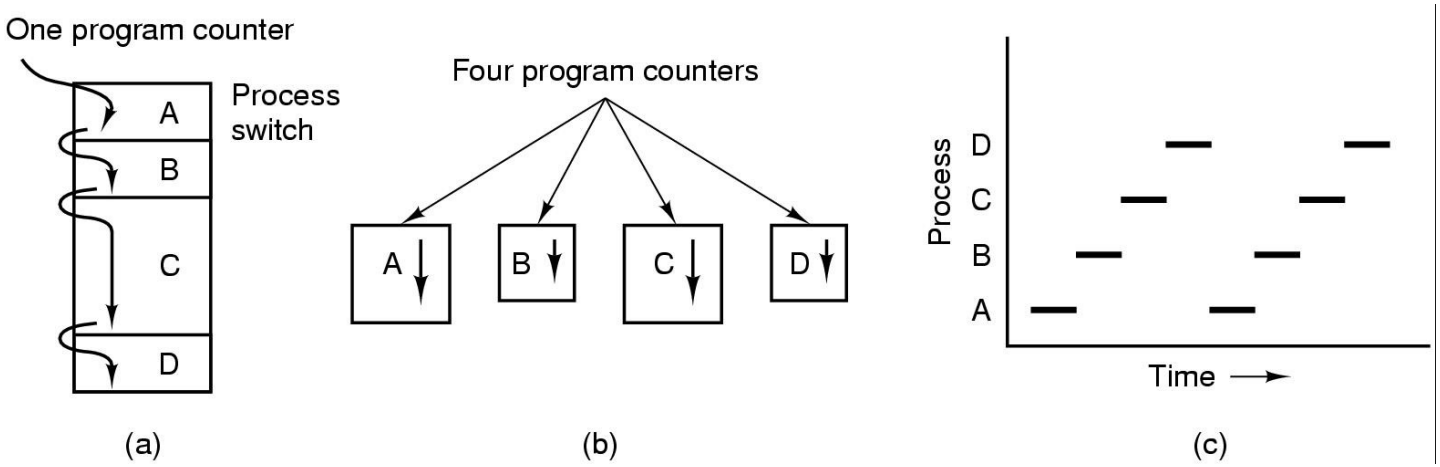
WIEDERHOLUNG - ENDE

Teil 5

- Was ist ein Rechnersystem?
- Was ist ein Betriebssystem?
- Aufgaben eines Systemadministrators
- Rechneraufbau
- Betriebssystemkonzepte
- Benutzer
- **Prozesse und Threads**

PROZESSE

Processes
The Process Model

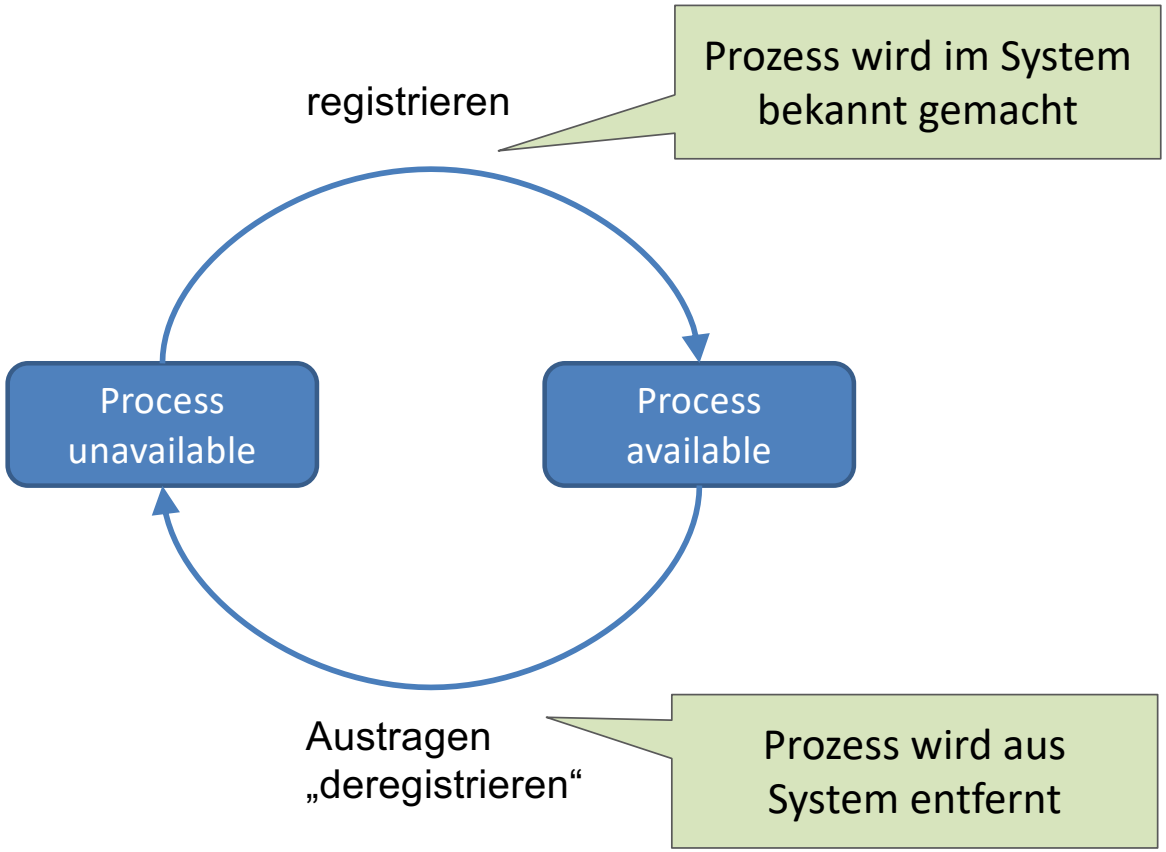


- Multiprogramming of four programs
- Conceptual model of 4 independent, sequential processes
- Only one program active at any instant

Wann werden Prozesse angelegt/erzeugt?

- Start des Systems
- Prozesserzeugung durch laufenden Prozess
  - Benutzeranforderung zum Start eines Prozesses
- …

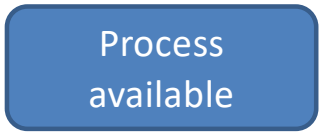
Prozess Lebenszyklus



Geschlossene versus offene Prozessmenge

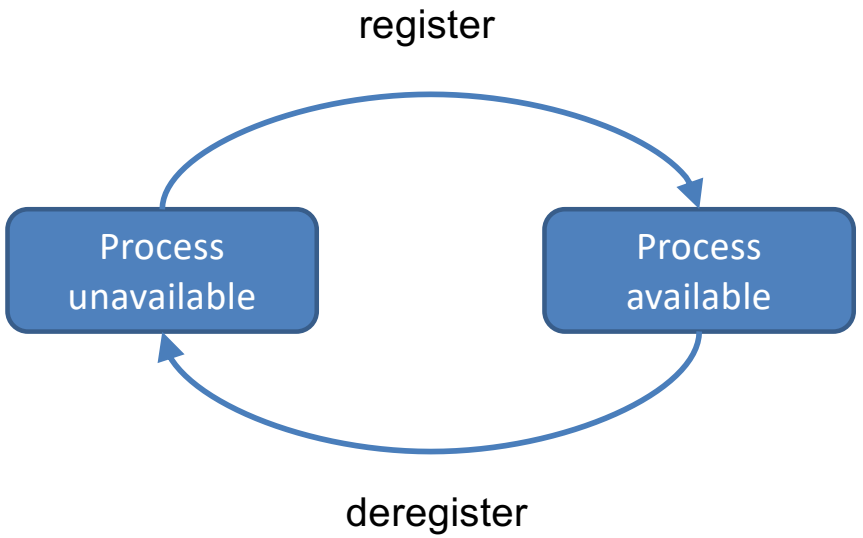
Geschlossen

zum Beispiel bestimmte eingebettete Systeme



Offen

zum Beispiel :UNIX, Windows



UNIX u. Windows: Prozesserzeugung

- Prozesserzeugung =
  1. Registrieren (Prozess im System bekannt machen bzw. anlegen)
  2. Aktivieren (Prozess bereit machen zur Ausführung ➔ Prozess nimmt am Scheduling teil)

- Systemdienst (aus der Standardbibliothek des Betriebssystems) zur Prozesserzeugung
  - UNIX: `fork`
  - Windows: `CreateProcess`

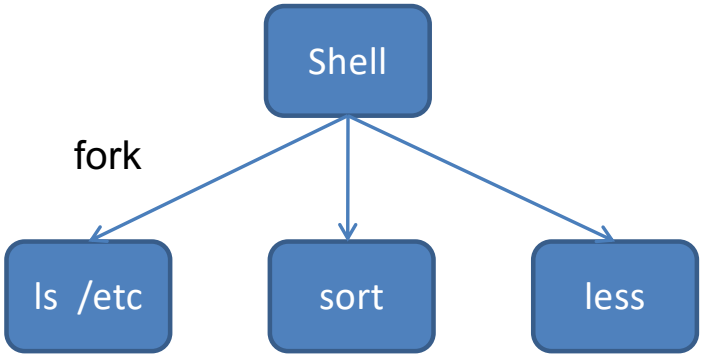
UNIX: fork & exec

fork

- Erzeugt Kindprozess (Child) mit gleichem Umfeld:
  - Programmcode
  - Speicherimage (Kopie)
  - Umgebungsvariablen (Kopie)
  - Offene Dateien

exec

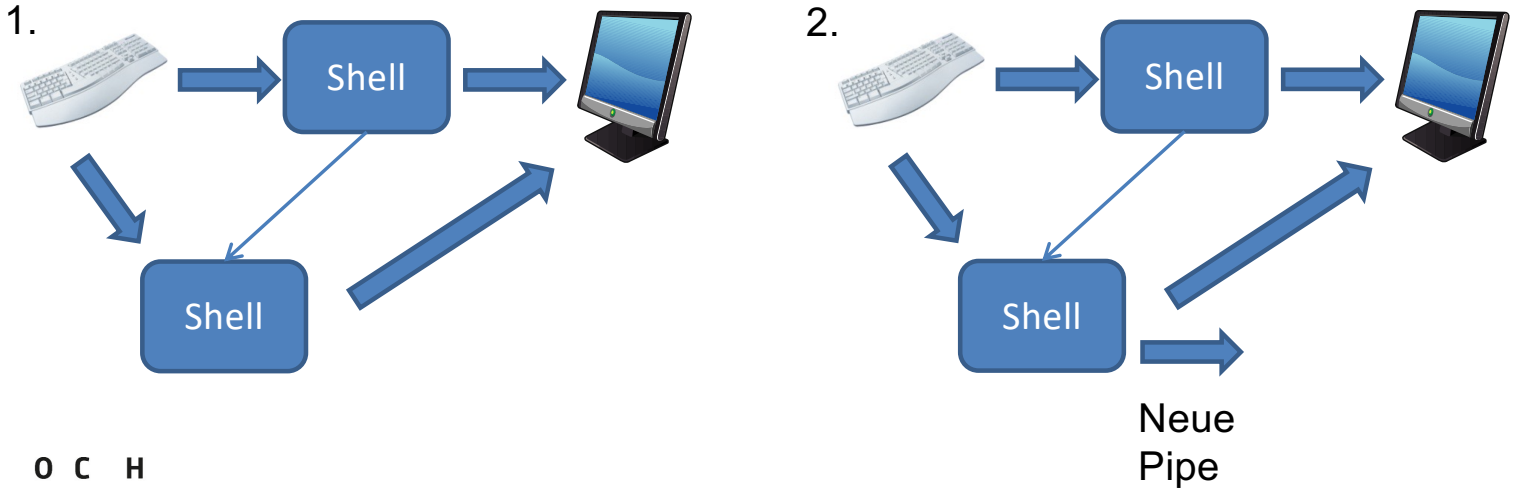
- Führt neues Programm anstelle des bisherigen aus



Beispiel: fork & exec (I)

„ls /etc | less“

1. fork (eine Seite von „|“)
2. Pipe erzeugen
3. fork (zweite Seite von „|“) und Pipe Enden zuordnen
4. 2 x exec („ls“, „less“)

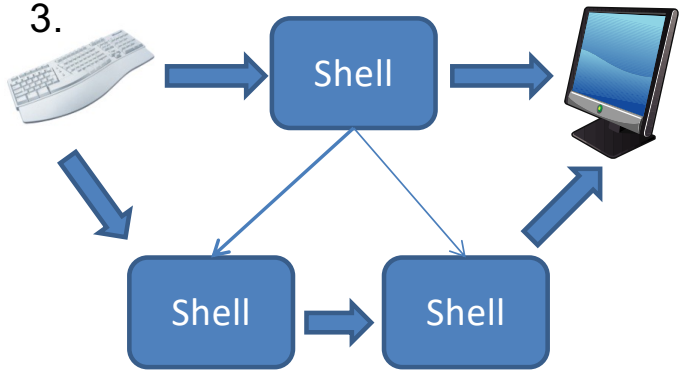


Beispiel: fork & exec (II)

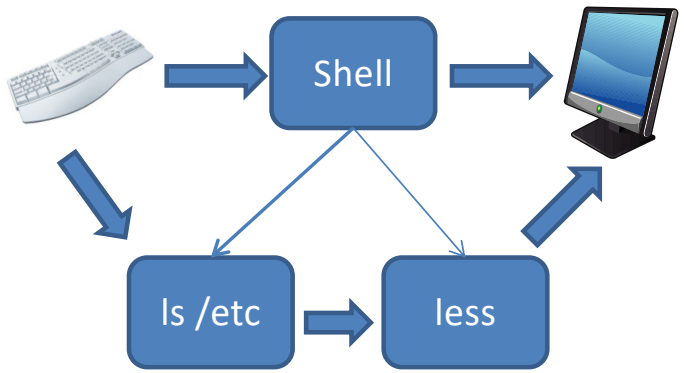
„ls /etc | less“

1. fork (eine Seite von „|“)
2. Pipe erzeugen
3. fork (zweite Seite von „|“) und Pipe Enden zuordnen
4. 2 x exec („ls“, „less“)

3.

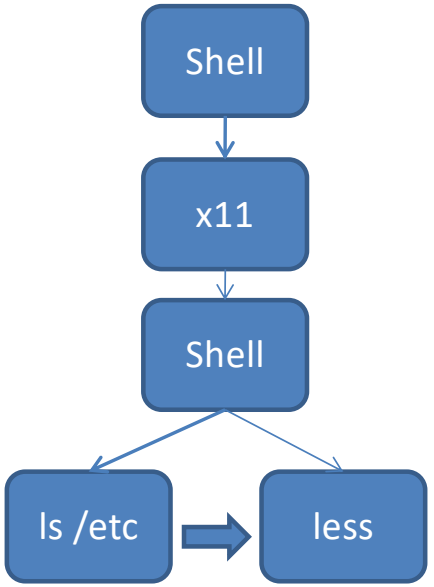


4.



Prozesshierarchie

- Elternprozess erzeugt Kindprozess, dieser kann wieder Kindprozess erzeugen, usw.
- Dies ergibt eine Hierarchie von Prozessen
  - Unter UNIX bezeichnet man dies als "process group"
- Windows kennt keine Prozesshierarchie



Wann werden Prozesse beendet?

- Normales Programmende
- Prozess beendet sich wegen Fehler
- System beendet Prozess wegen Fehler
- Kill durch anderen Prozess (z.B. bei shutdown)
- …

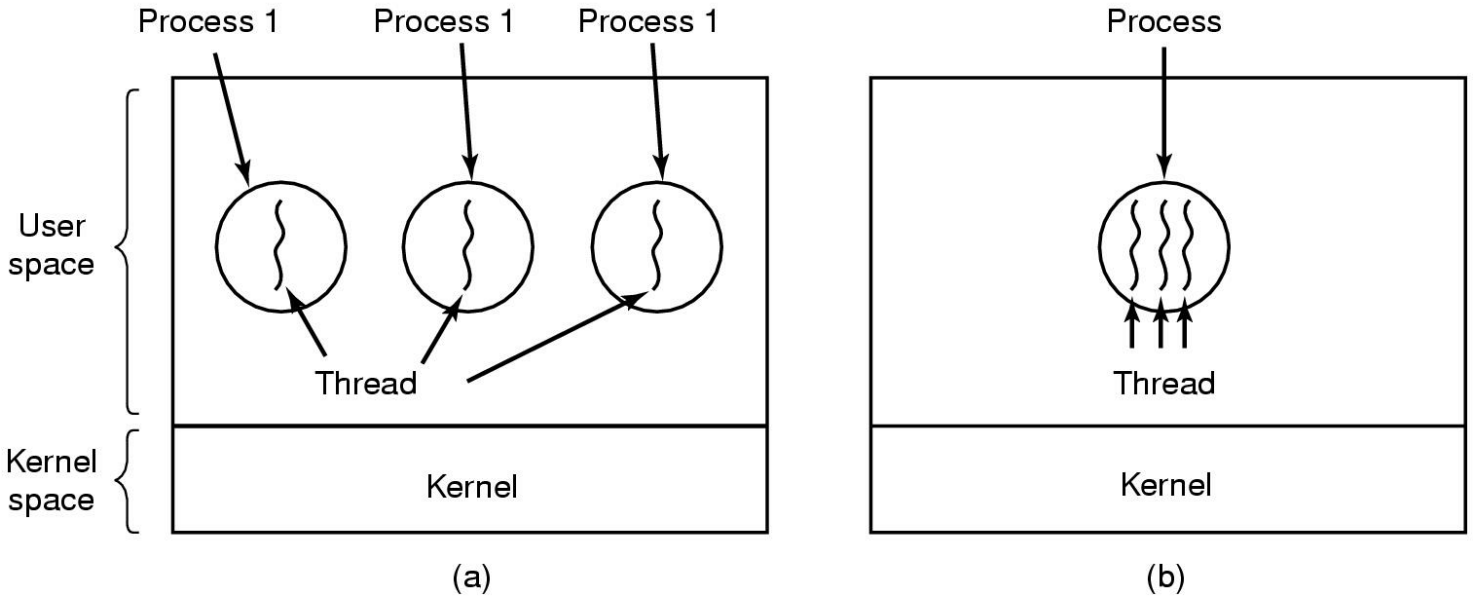
UNIX u. Windows: Prozessende

- Prozessende =
  1. terminate (Prozessausführung wird beendet)
  2. deregister (Prozess wird aus System entfernt)

- System Calls zur Prozessbeendigung von „außen“
  - UNIX: `kill`
  - Windows: `TerminateProcess`

THREADS

Threads
The Thread Model (1)



(a) Three processes each with one thread

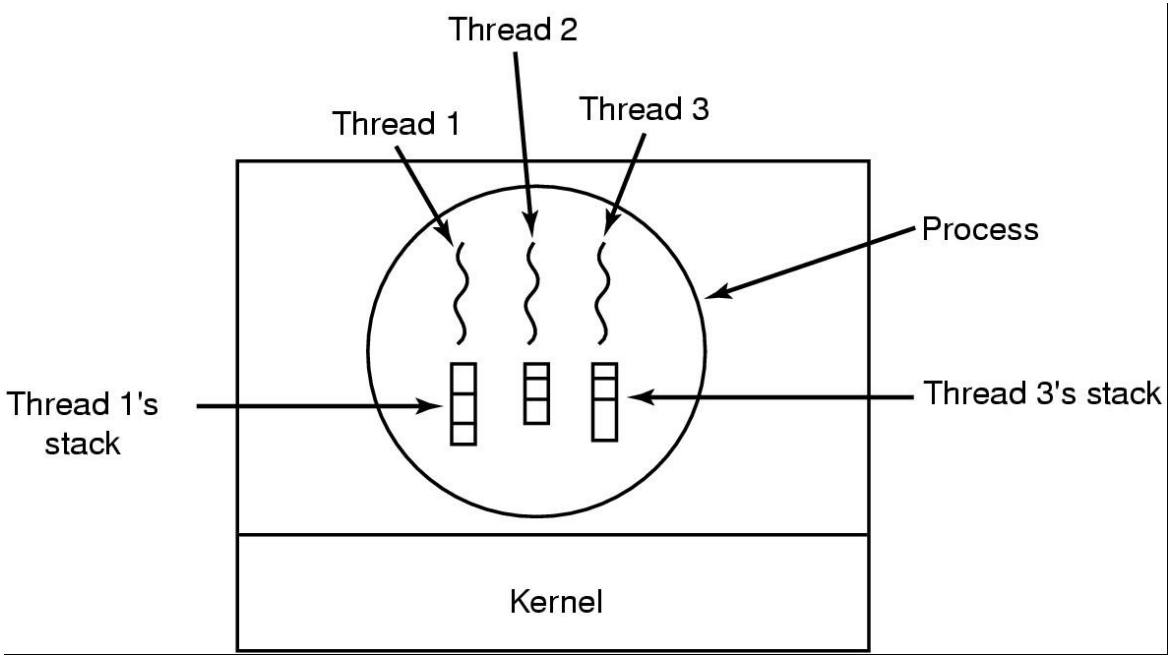
(b) One process with three threads

The Thread Model (2)

Per process items	Per thread items
Address space	Program counter
Global variables	Registers
Open files	Stack
Child processes	State
Pending alarms	
Signals and signal handlers	
Accounting information	

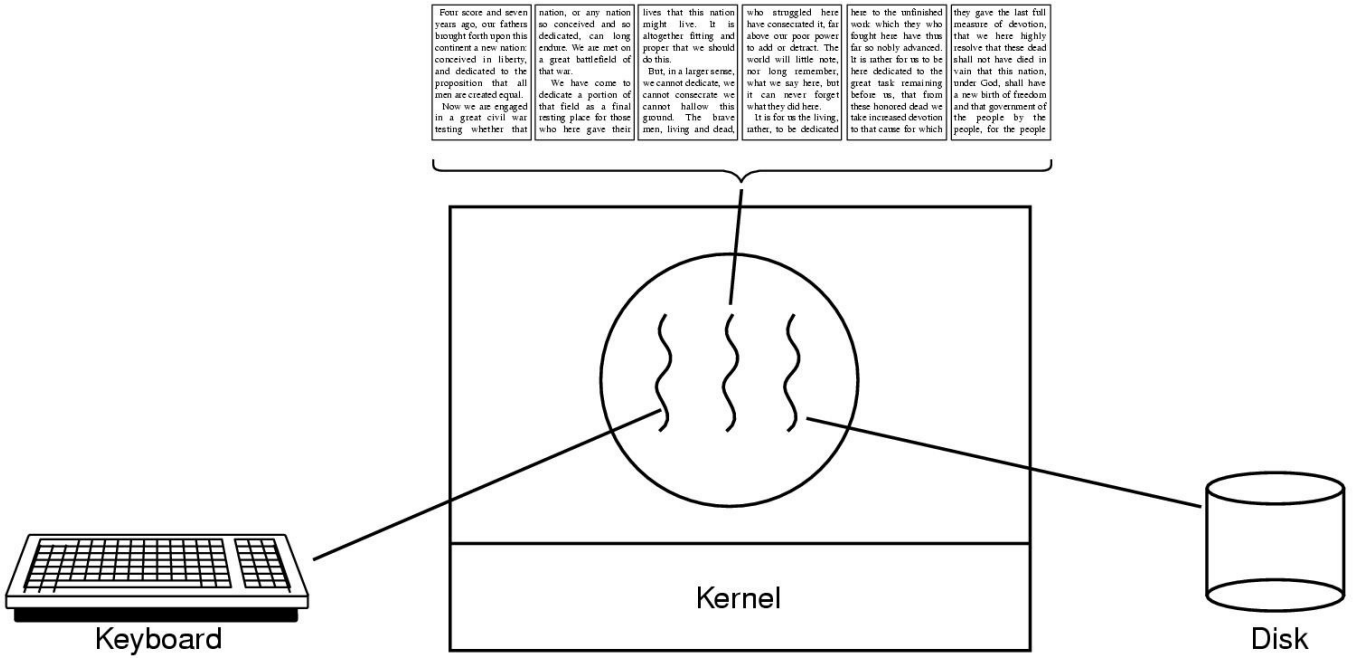
- Items shared by all threads in a process
- Items private to each thread

The Thread Model (3)



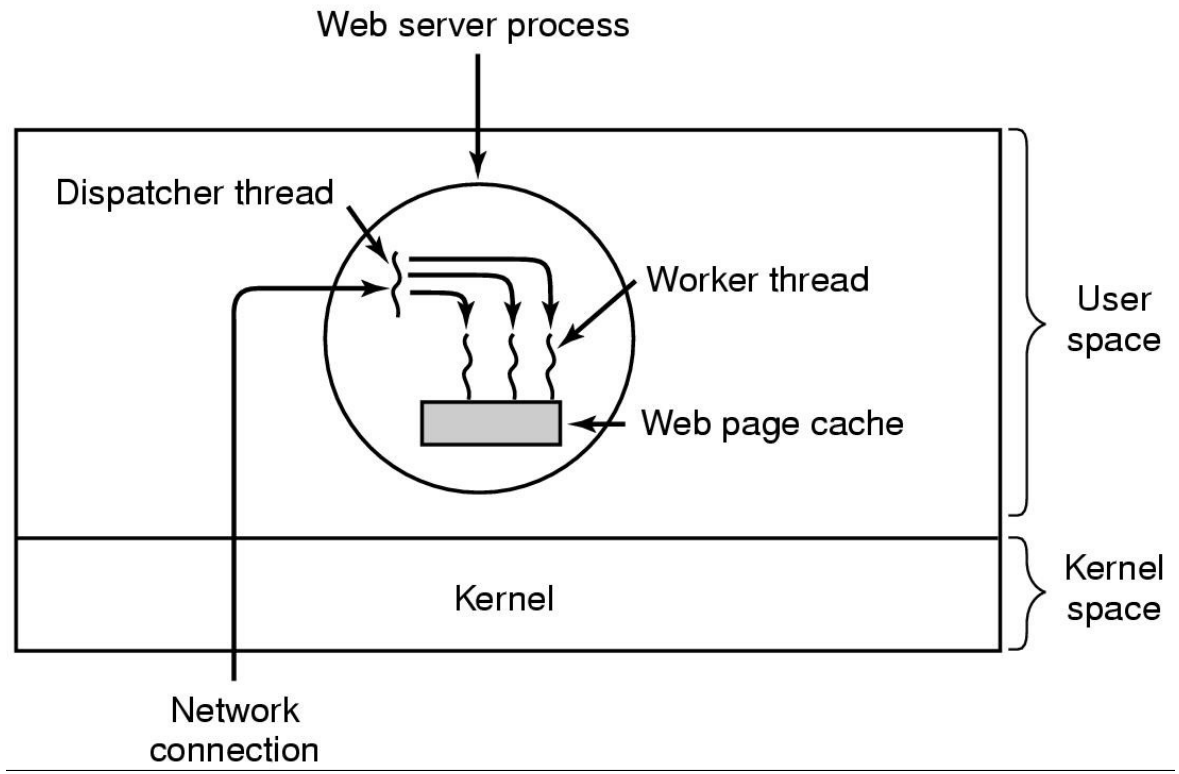
Each thread has its own stack

Thread Usage (1)



A word processor with three threads

Thread Usage (2)



A multithreaded Web server