03. Process

Created	@February 17, 2022 7:07 PM
Tags	 Lee Edith.D

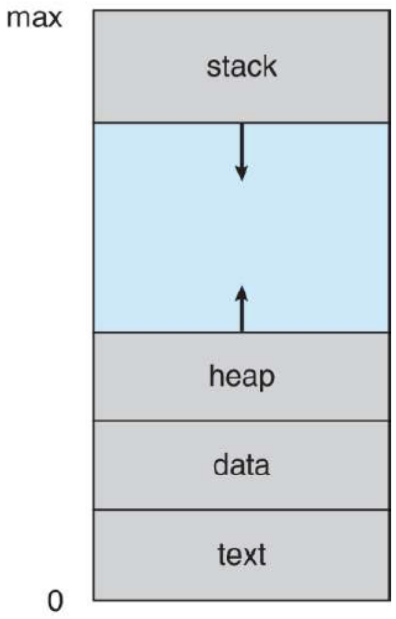
Backgrounds

- a program in execution
- owns the entire computer

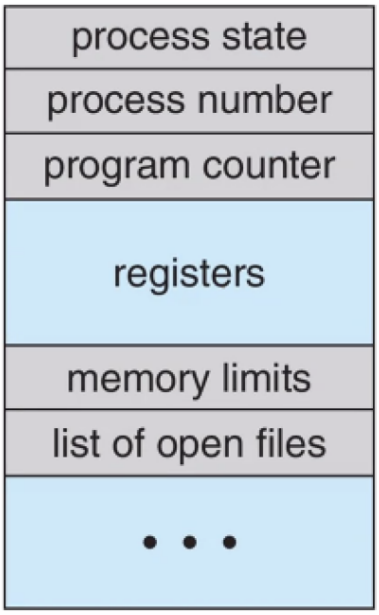
Process Store

Concepts

- how process stored in memory
  - consists of
    - text section / code section
    - PC
    - stack: contains temporary data (function args...)
    - data section: contains constant, variables
    - heap: allocate memory dynamically



- Process state
  - new
  - running
  - waiting
  - ready
  - terminate
- PCB
  - a data structure
  - consists
    - process state
    - process number: **pid**
    - program counter
    - registers:
    - memory limits
    - list of open files



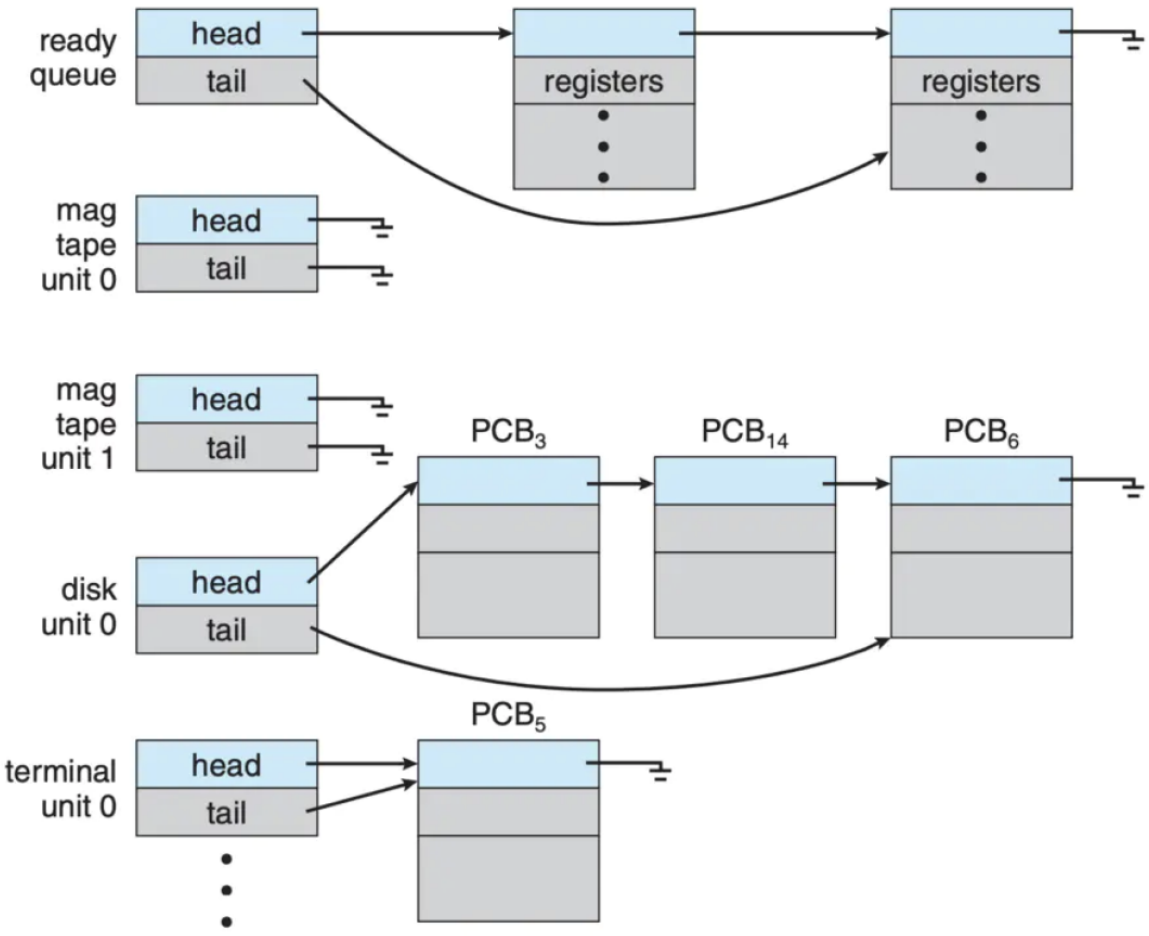
Process in Linux

- **`task_struct`** in **`<linux/sched.h>`**

```
pid t pid; /* process identifier */
long state; /* state of the process */
unsigned int time slice /* scheduling information */
struct task_struct *parent; /* this process's parent */
struct list_head children; /* this process's children */
struct files_struct *files; /* list of open files */
struct mm_struct *mm; /* address space of this pro */
```

Process Scheduling

Queue ← Link Method



- job queue
  - when a process get into a system, it is added to job queue
- ready queue
  - contains all ready processes waiting to be executed
- device queue
  - shared device has a queue to store processes
- job queue and ready queue are implemented by a **link**
  - 2 pointers → PCB
    - head
    - tail

CPU Scheduling

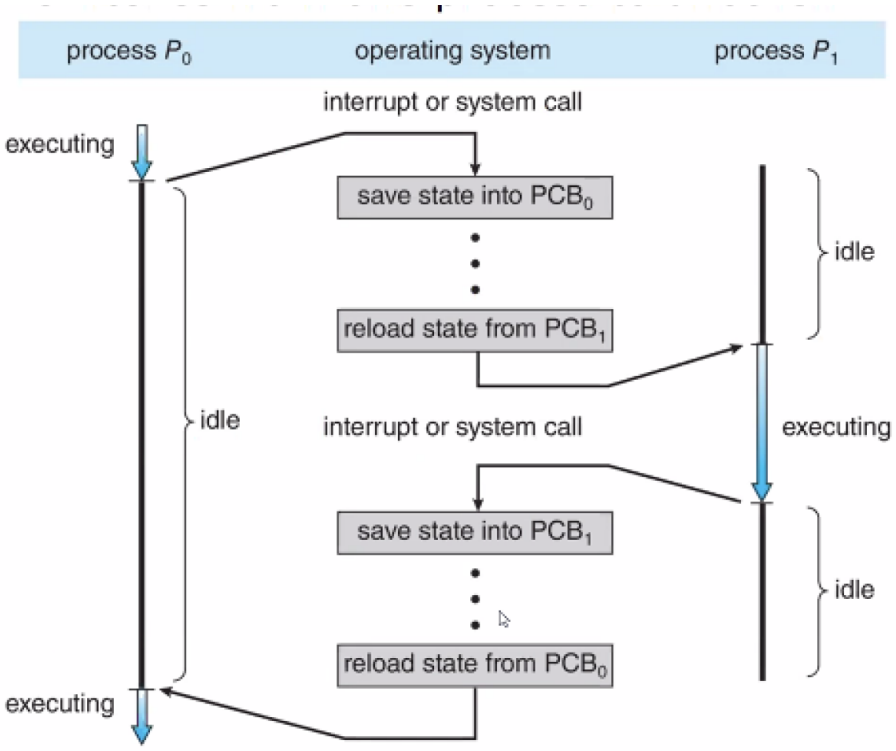
- process enter a system
  - add to ready queue
  - state: ready
- process enter real CPU
  - execute
- process need to visit devices when executing
  - add to device queues
  - state: waiting

Scheduling Program

- process scheduling is done by scheduling program
- generally, in a system need to execute a lot of processes
  - processes can’t be allocated all to real memory
  - stored in disk → virtual memory
- 长期调度程序：会把在磁盘上的进程加载到内存，把最近不会被执行的进程从内存中挪出。
  - I/O密集型进程
  - CPU密集型进程。
- 短期调度程序(CPU调度程序)：从就绪队列中选择进程分配CPU。

Process Switch （上下文切换）

- save old state
- load new state



Process Management

Process Set-up

- `fork()`
  - parent process generate children processes, forming a tree of processes
  - resources in child process
    - from OS (no share)
    - from parent (share)
  - execution
    - concurrent
    - parent waits until children terminate
  - return pid
- `pid` $= 0$ child

- `pid` > 0 parent

Process Execution

- `exec()`
  - load new .exe file into child PCB to execute
  - parent enters waiting queue

Process Terminates

- `exit()`
  - child calls it and ask OS to delete itself.
  - return parent
    - if parent is waiting, all resources will be released by OS
    - `wait()` contains process table
- others
  - only can be terminated by parent
    - child resources overflow
    - no need for child's tasks
    - parent is terminating
- 僵尸进程
  - child terminates
  - parent not waiting
- 孤儿进程
  - child terminates
  - parent terminates without waiting

Status

User Status

Kernel Status

Files, IPC (Interprocess Cooperation)

- Aim
  - information sharing
  - computation speedup
  - modularity
  - convenience
- 2 modules
  - shared memory
  - message passing

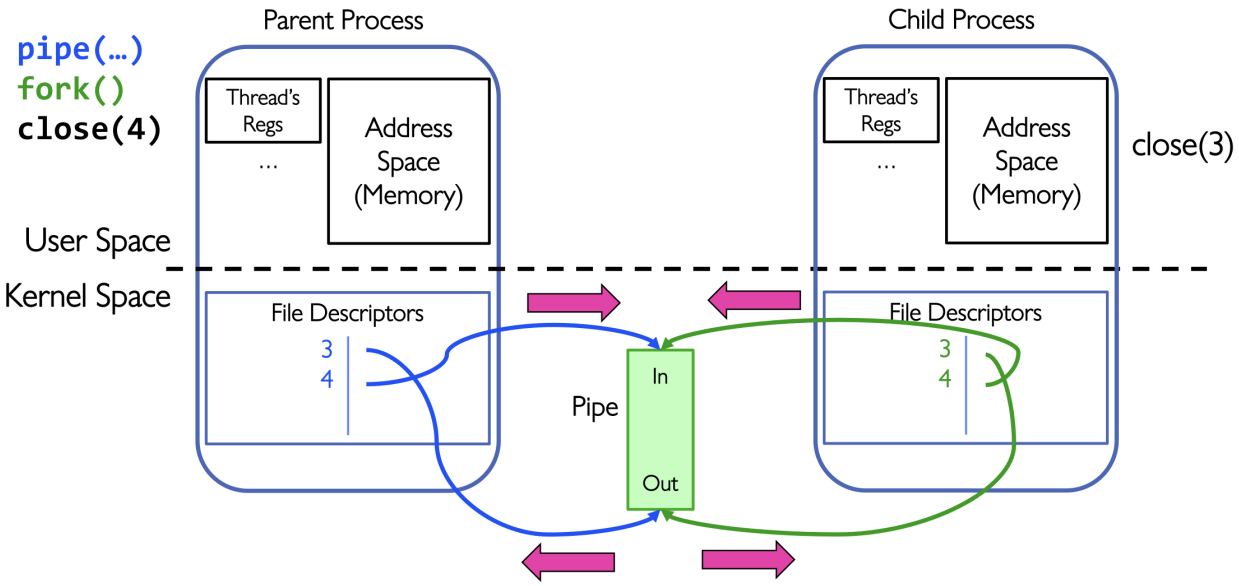
Shared Memory

- process

Message Passing

- common in distributed system

Pipe



Socket

- consists:
  - IP
  - port
    - each process needs a port
- RPC (Remote-Process-Communication)