操作系统原理与设计

第3章 Processes（进程）2

陈香兰

中国科学技术大学计算机学院

2009年09月01日

提纲

Outline

Process Scheduling

The objective of multiprogramming: ???

The objective of time sharing: ???

What the system need?

the process scheduler selects an available process to execute on the CPU.

Process Scheduling

The objective of multiprogramming: ???

The objective of time sharing: ???

What the system need?

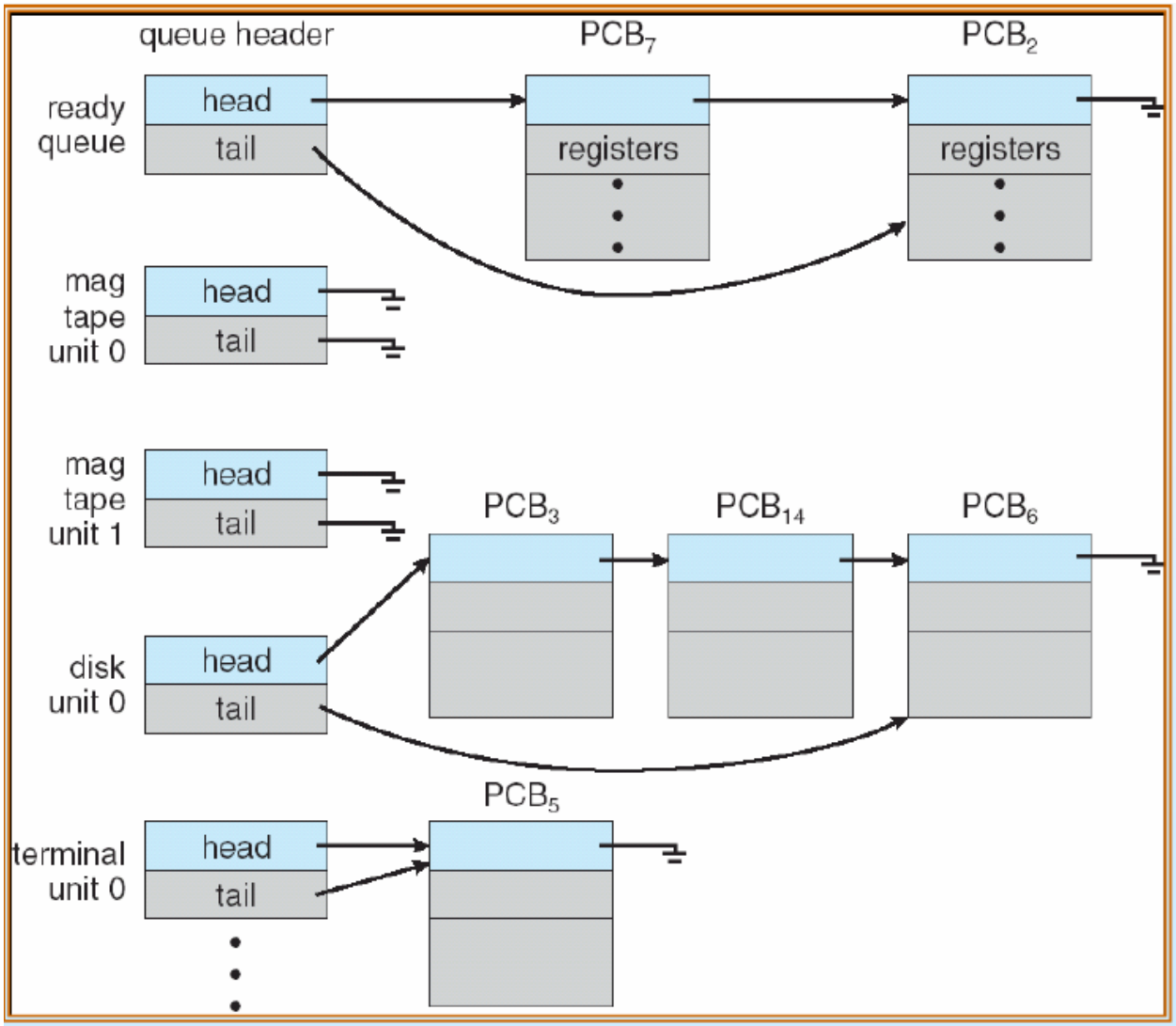
the process scheduler selects an available process to execute on the CPU.

Process Scheduling Queues

Processes migrate among the various queues

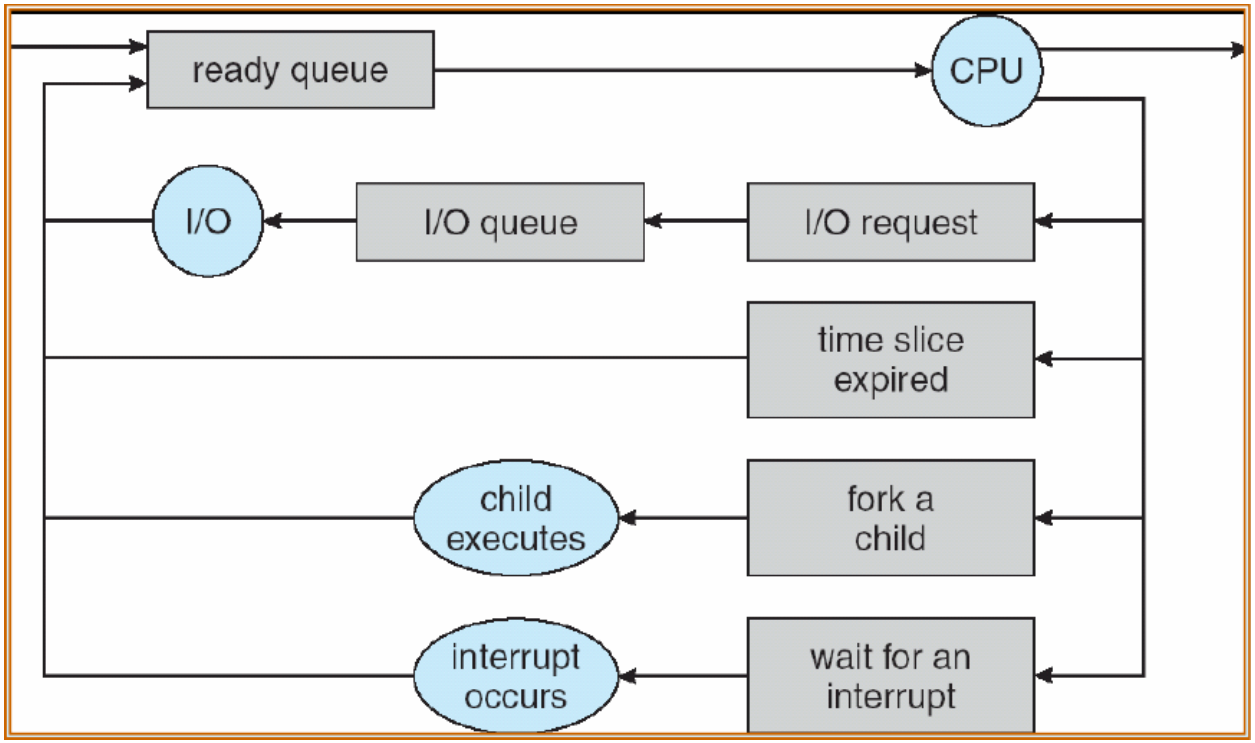
- **Job queue** – set of all processes in the system
- **Ready queue** – set of all processes residing in main memory, ready and waiting to execute
- **Device queues** – set of processes waiting for an I/O device

Ready Queue And Various I/O Device Queues



Representation of Process Scheduling

- a queueing diagram



Outline

Schedulers I

Long-term scheduler (or job scheduler)

- selects which processes should be brought into the ready queue

Short-term scheduler (or CPU scheduler)

- selects which process should be executed next and allocates CPU

The primary distinction between long-term & short-term schedulers I

- The primary **distinction** between long-term & short-term schedulers lies in **frequency of execution**
  - Short-term scheduler is invoked very frequently (UNIT: ms) (must be fast)
  - Long-term scheduler is invoked very infrequently (UNIT: seconds, minutes) (may be slow)
  - WHY?
- The long-term scheduler controls **the degree of multiprogramming**
  - the number of processes in memory.
  - stable?

The primary distinction between long-term & short-term schedulers II

- Processes can be described as either:

I/O-bound process

- spends more time doing I/O than computations, many short CPU bursts

CPU-bound process

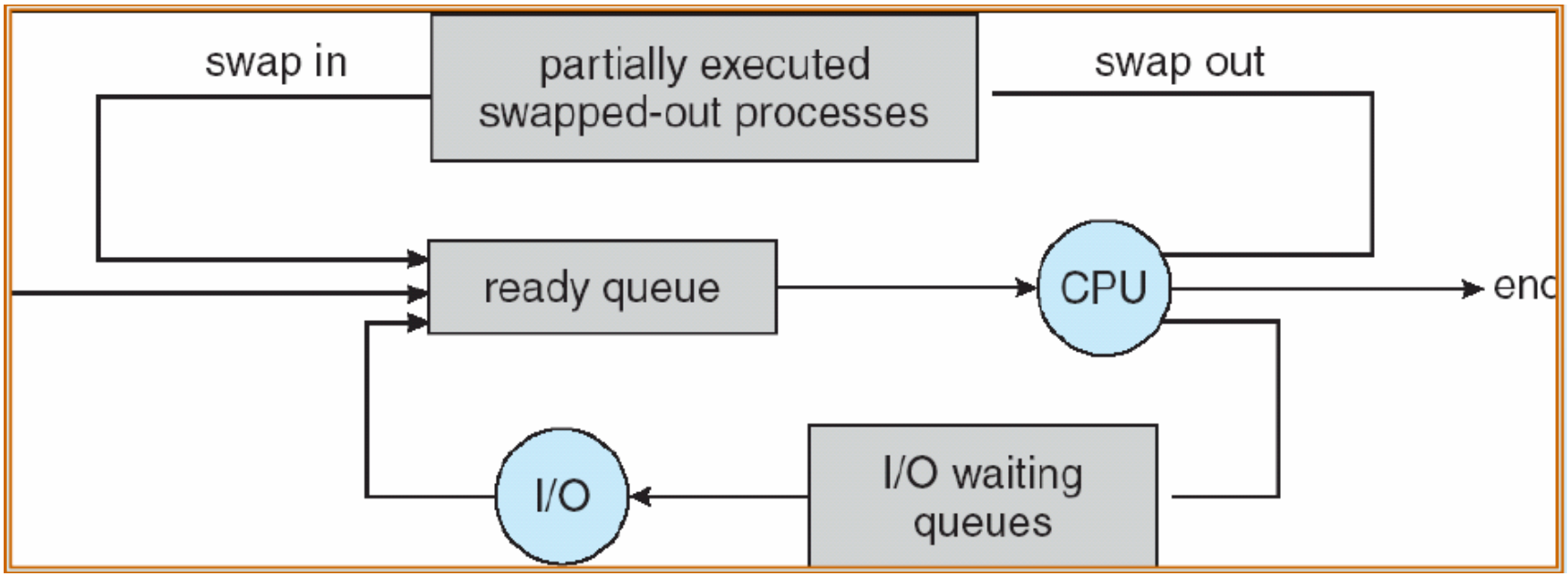
- spends more time doing computations; few very long CPU bursts

- **IMPORTANT** for long-term scheduler:
  - A good process mix of I/O-bound and CPU-bound processes.

The primary distinction between long-term & short-term schedulers III

- The long-term scheduler may be absent or minimal
  - UNIX, MS Windows, ...
  - The stability depends on
    - physical limitation
    - self-adjusting nature of human users

Addition of Medium Term Scheduling



- swapping

Outline

Context Switch I

- **CONTEXT**
  - when an interrupt occurs; When scheduling occurs

PCB

- CPU registers
- process state
- memory-management info
- ...

- operation: **state save** VS. **state restore**

Context Switch II

- Context switch
  - When CPU switches to another process, the system must save the state of the old process and load the saved state for the new process
  - Context-switch time is **overhead**; the system does no useful work while switching
  - Time dependent on hardware support (typical: n μs
    - CPU & memory speed
    - *N* of registers
    - the existence special instructions

Code reading

- 观察
  - 队列的组织
  - 上下文的内容和组织
  - 上下文切换
- linux-0.11
- linux-2.6.26
- uC/OS-II

Outline

Process Creation I

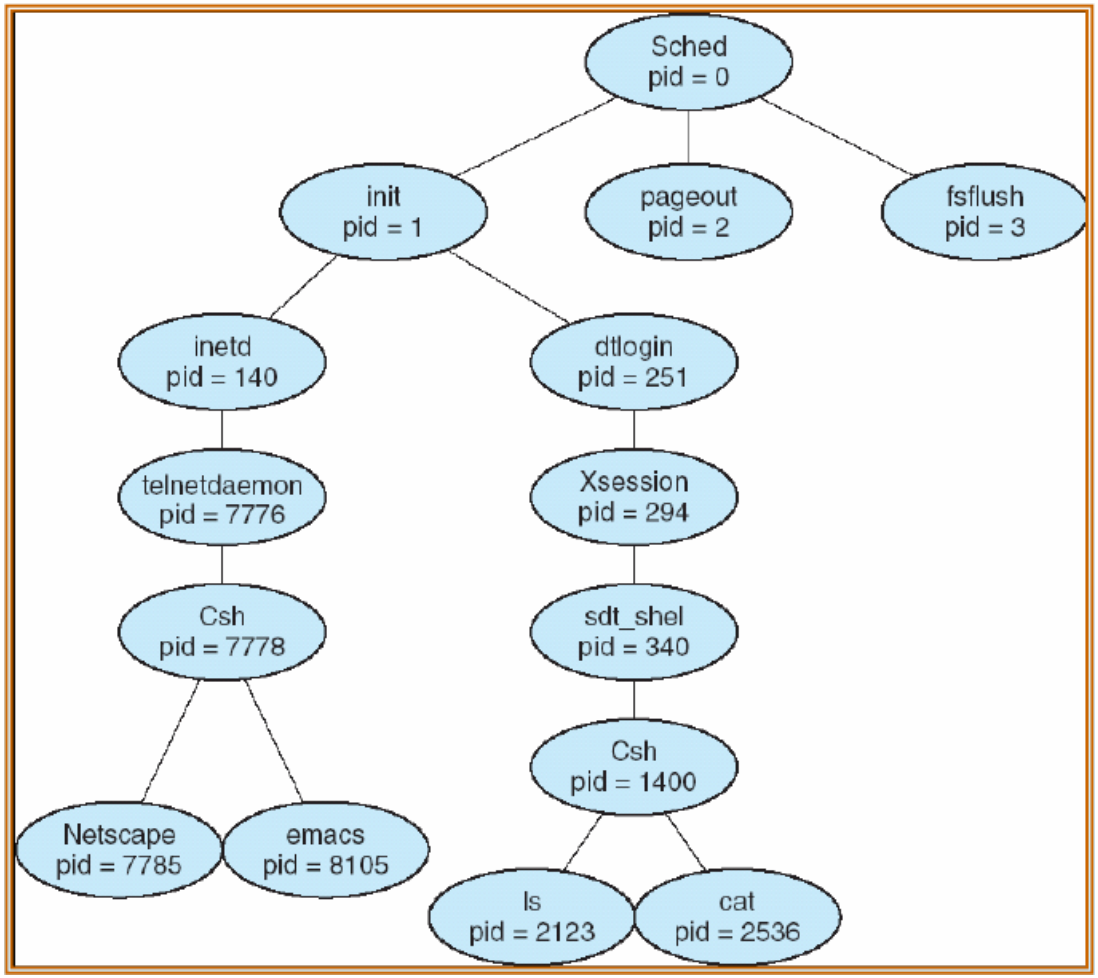
- **Parent process** create **children processes**, which, in turn create other processes, forming a **tree of processes**
- pid
- UNIX & Linux

Command:

ps -el

Process Creation II

- A tree of processes on a typical Solaris

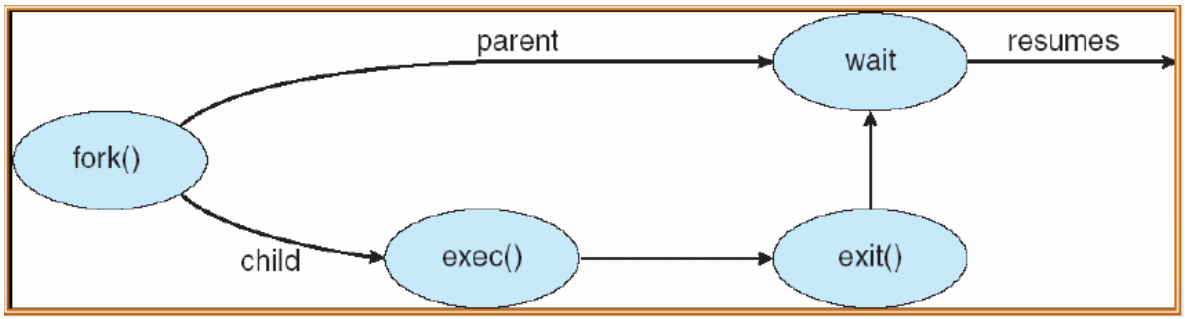


Process Creation III

- **Resource sharing**
  - Parent and children share all resources
  - Children share subset of parent’s resources
  - Parent and child share no resources
- **Execution**
  - Parent and children execute concurrently
  - Parent **waits** until children terminate
- **Address space**
  - Child duplicate of parent
  - Child has a program loaded into it

Process Creation IV

- UNIX examples: fork + exec
  - **fork** system call creates new process
  - **exec** system call used after a **fork** to replace the process' memory space with a new program



C Program Forking Separate Process

```
int main(void) {
    pid_t pid;
    /* fork another process */
    pid = fork();
    if (pid < 0) {      /* error occurred */
        fprintf(stderr, "Fork Failed");
        exit(-1);
    } else if (pid == 0) {      /* child process */
        execlp("/bin/ls", "ls", NULL);
    } else {      /* parent process */
        /* parent will wait for the child to complete */
        wait (NULL);
        printf ("Child Complete");
        exit(0);
    }
}
```

Outline

Process Termination I

- Process executes last statement and asks the operating system to delete it (exit)
  - Output data from child to parent (via wait)
  - Process' resources are deallocated by operating system
- Parent may terminate execution of children processes (abort)
  - Child has exceeded allocated resources
  - Task assigned to child is no longer required
  - If parent is exiting

Some operating system do not allow child to continue if its parent terminates

- All children terminated - cascading termination

小结

作业

- 名词解释：
  - 长 / 中 / 短期调度
  - 多道程序度
  - IO密集型 / CPU密集型

华夏班上机作业

- 基于i386的模拟器（推荐bochs）
  - 编写启动加载程序。（从16位实模式启动，加载OS代码，进入32位保护模式，能调用C语言编写的OS入口程序）
  - 编写任务管理和上下文切换代码
    - 能够从OS入口程序主动切换到被创建的任务上

非华夏班上机作业

- 在ftp://alpha.gnu.org/gnu/grub/上下载grub-0.97.tar.gz，编译
- 制作grub启动软盘
- 到网络上下载一个可用的OS映像，编写menu.lst，利用grub启动之
  - dlxlinux，或其他
  - 这里提供2个RTEMS操作系统的映像
    - hello.exe
    - ticker.exe

谢谢！