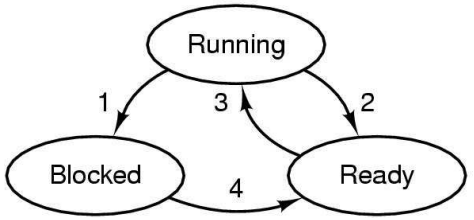
Scheduling

Ho più di un processo in stato *ready*

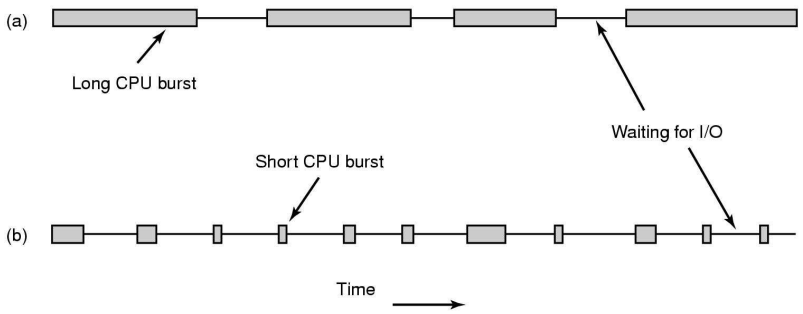
Quale scelgo?

Problema dello *scheduling*



1. Process blocks for input
2. Scheduler picks another proc
3. Scheduler picks this process
4. Input becomes available

Periodi di uso della CPU alternati ad attese per I/O



(a) CPU bound

(b) I/O bound

Obiettivi

Tutti i sistemi
- fairness (equità)
- policy enforcement
- max. CPU and I/O utilization

Sistemi a lotti
- max. throughput
- min. turnaround
- max. CPU utilization

Sistemi interattivi
- min. tempi di risposta

Sistemi real-time
- rispettare le deadline

Quando viene eseguito lo scheduling?

1. creazione/distruzione di un nuovo processo
2. un processo si blocca per fare I/O
3. arriva un interrupt per I/O
4. clock interrupt
5. all'uscita da una system call

Due tipi di scheduling

Senza prelazione (non-preemptive):
un processo esegue fino a che non cede volontariamente la CPU

Con prelazione (preemptive):
un processo in esecuzione può essere bloccato dal S.O.

Scheduling in batch systems I

First come, first served (senza prelazione)

Il processo esegue fino a che non si blocca; poi torna in fondo alla coda

- facile da implementare
- problema: un processo CPU bound rallenta 100 processi I/O bound

Scheduling in batch systems II

Shortest job first (senza prelazione)

Presumiamo di conoscere in anticipo la durata dei processi

Come *first come, first served* ma la coda è ordinata per durata crescente

Shortest job first

(i processi vengono eseguiti da destra a sinistra)

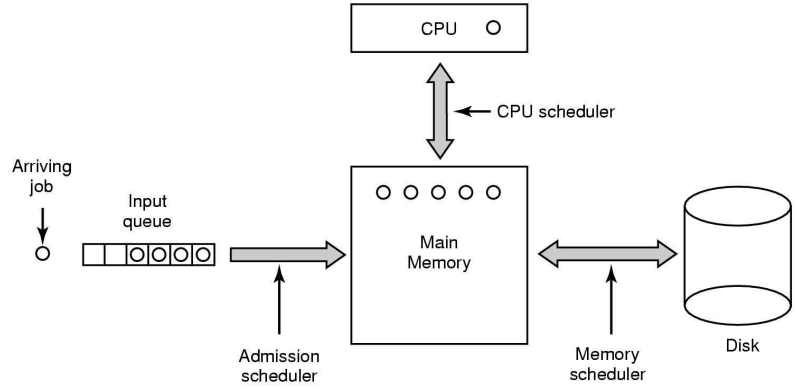
average turnaround time

$$(a) \quad \frac{4+8+12+20}{4} = 11$$

$$(b) \quad \frac{8+12+16+20}{4} = 14$$

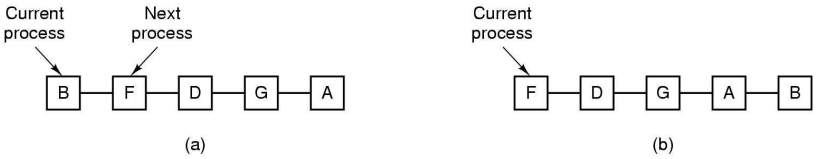
Scheduling a tre livelli nei sistemi a lotti

0. admission scheduler
1. memory scheduler
2. CPU scheduler



Scheduling in interactive systems

Round-robin



lista di processi *ready*

Quando il processo esaurisce il *quanto* torna in fondo alla lista

Priority scheduling

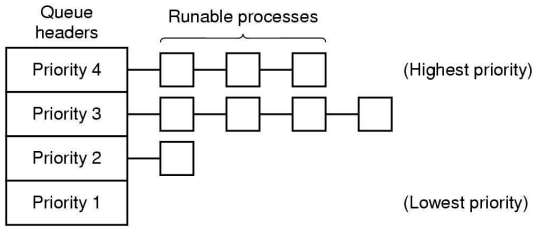
Assegnamo una priorità ai processi

Scegliamo sempre il processo di priorità massima

Problema: starvation $\Rightarrow$ priorità dinamiche

- privilegiare i processi I/O bound

Priority classes



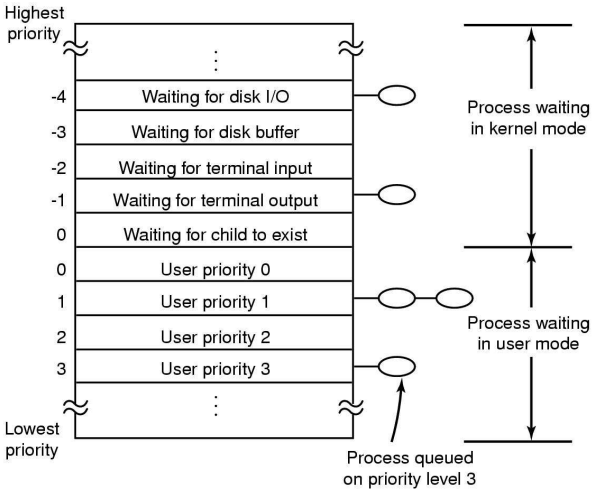
Policy versus Mechanism

Il kernel implementa un meccanismo di scheduling parametrizzabile

I processi possono modificare i parametri (policy)

Scopo: ottimizzare la performance conoscendo il comportamento dei processi

Scheduling in UNIX



Scheduling in UNIX

Code multiple ordinate per *priorità*

algoritmo round-robin all'interno della coda

quanto di tempo

UNIX: Calcolo della priorità

ricalcolata ogni secondo

priority = CPU_usage + nice + base

CPU_usage
- incrementato ad ogni clock tick
- dimezzato a intervalli di 1sec

nice
- range -20 – +20
- default value 0
- solo *root* può assegnare valori negativi

base
- serve a dare priorità ai proc. in kernel mode

Therefore we have arranged the whole system as a society of sequential processes, progressing with un- defined speed ratios. To each user program accepted by the system corresponds a sequential process, to each input peripheral corresponds a sequential process, to each output peripheral corresponds a sequential process.

... This enabled us to design the whole system in terms of these abstract sequential processes. delaying the progress of a process temporarily can never be harmful to the interior logic of the process delayed.

— Dijkstra, 1968

Scheduling nei sistemi real-time

Distinguiamo

- Hard real-time
- Soft real-time

Eventi periodici e aperiodici

Il sistema è schedulabile?

Scheduler dinamici o statici

La struttura interna di Minix 3