

Sistemas Operativos

LEI - 2019/2020

:: Gestão de Processos ::

Escola Superior de Tecnologia de Setúbal - IPS

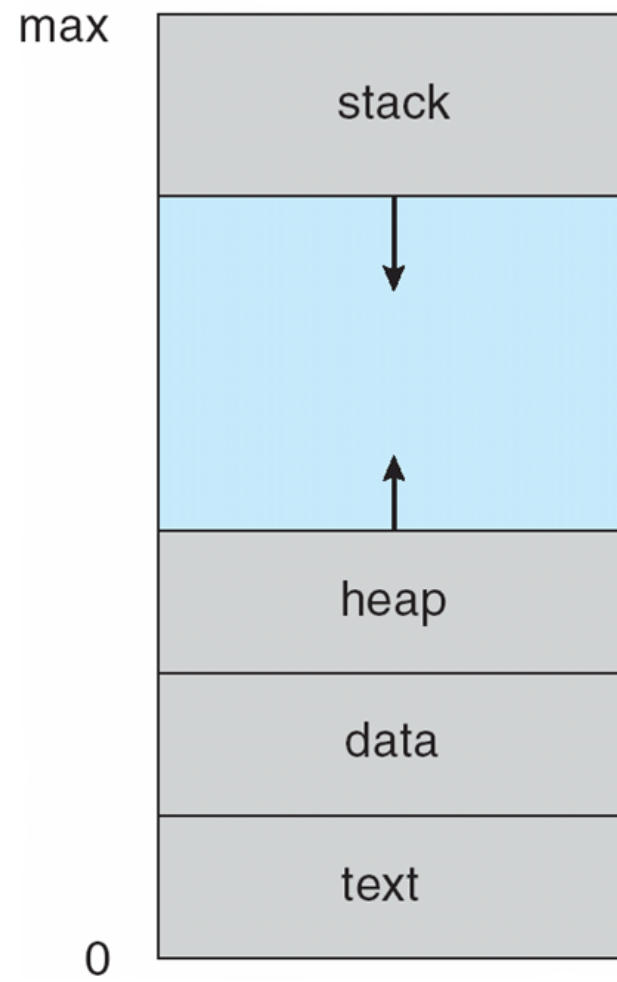
Conteúdos

- Noção de processos
- Características de processos (escalonamento, criação e terminação)
- Comunicação entre processos (IPC)

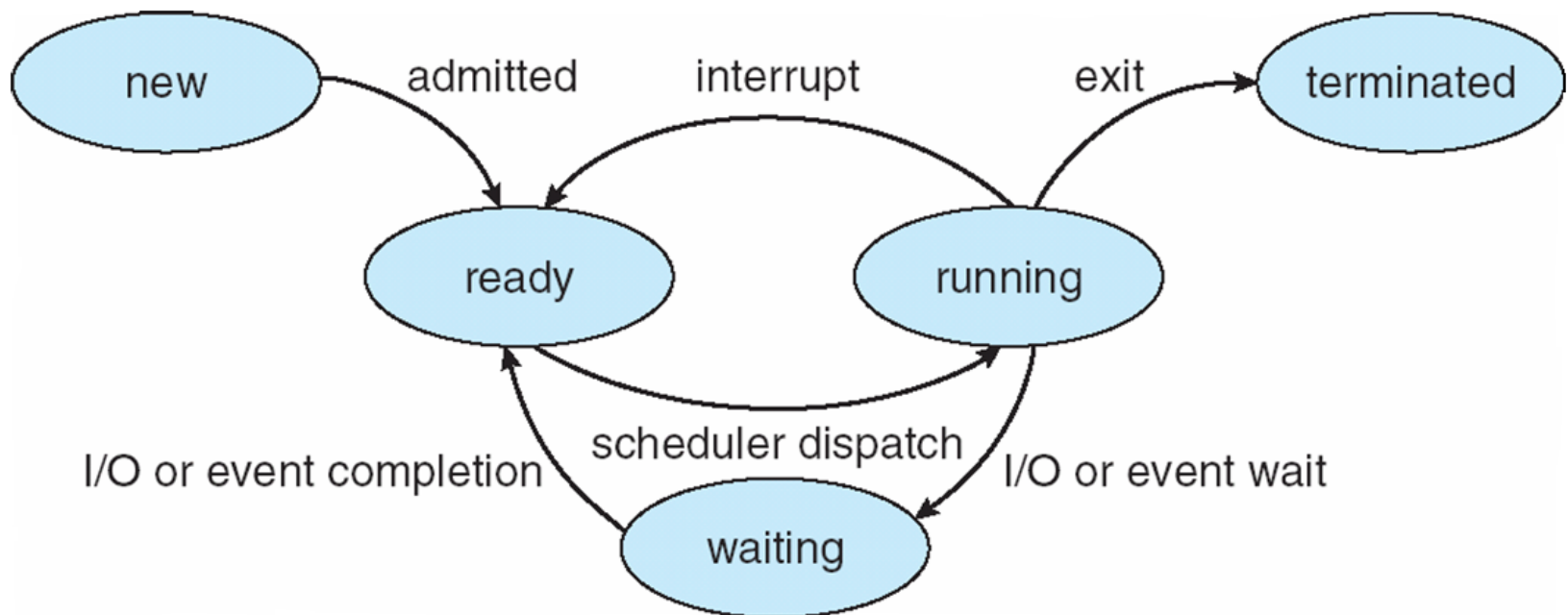
Processo

Programa em execução....

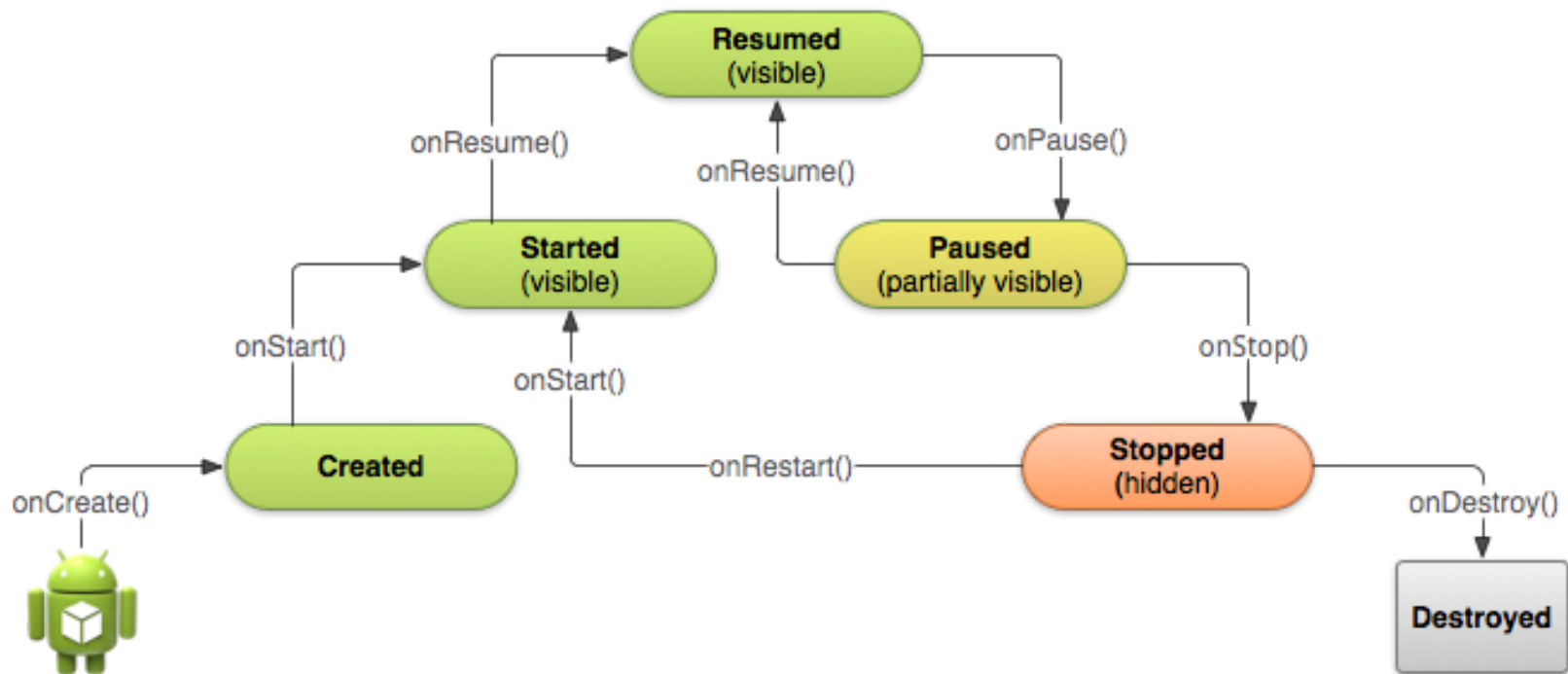
- Text section (código)
- Data section (variáveis globais)
- Heap
- Stack
- Registos (program counter, etc.)



Estado de um processo



Ex: Android Activity lifecycle

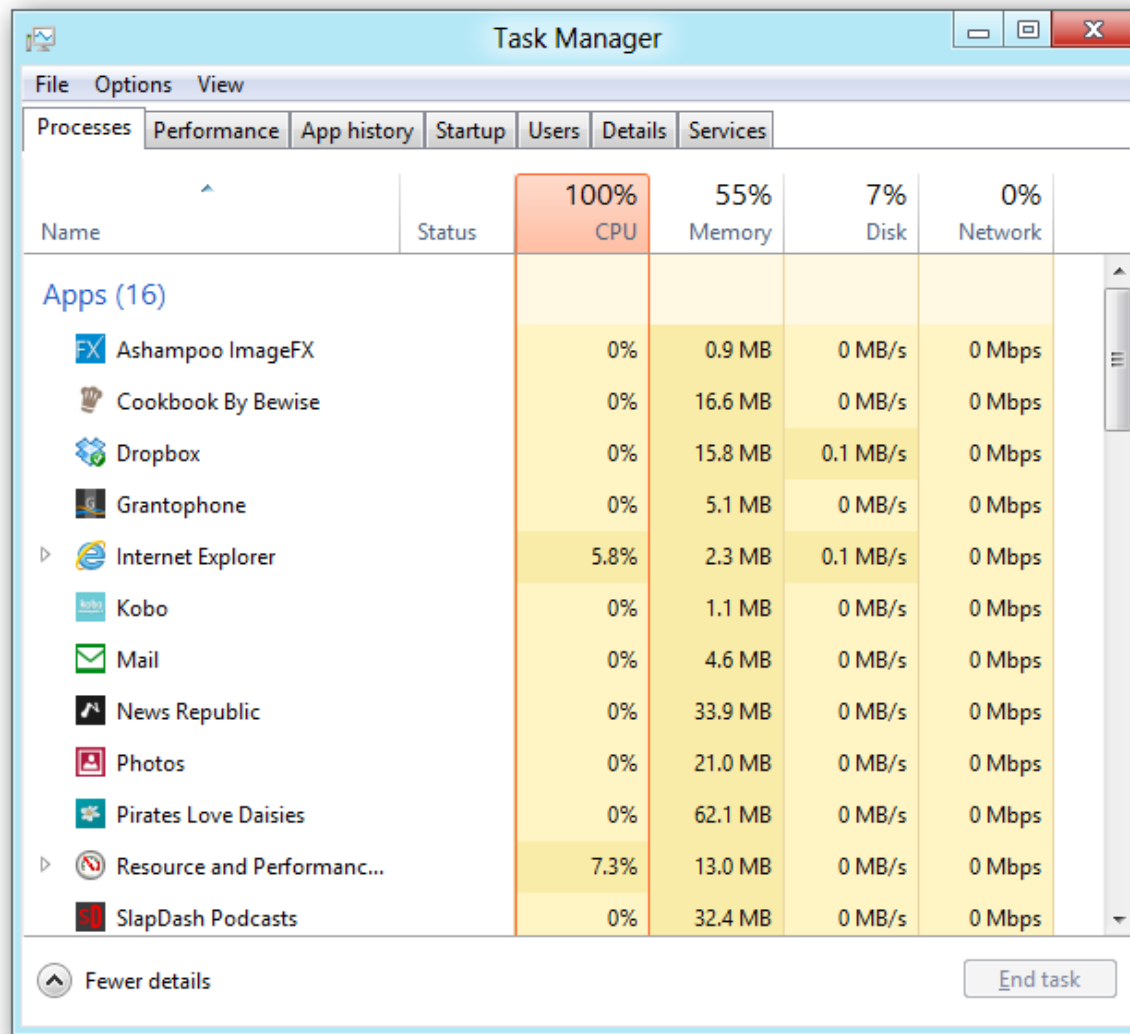


Linux (ps)

```
Terminal - jventura@jventura-VirtualBox: ~
File Edit View Terminal Tabs Help

jventura@jventura-VirtualBox:~$ ps
  PID TTY          TIME CMD
 1609 pts/0        00:00:00 bash
 1726 pts/0        00:00:00 ps
jventura@jventura-VirtualBox:~$ ps -A
  PID TTY          TIME CMD
   1 ?            00:00:01 systemd
   2 ?            00:00:00 kthreadd
   3 ?            00:00:00 kworker/0:0
   4 ?            00:00:00 kworker/0:0H
   5 ?            00:00:00 kworker/u2:0
   6 ?            00:00:00 mm_percpu_wq
   7 ?            00:00:00 ksoftirqd/0
   8 ?            00:00:00 rcu_sched
   9 ?            00:00:00 rcu_bh
  10 ?            00:00:00 migration/0
  11 ?            00:00:00 watchdog/0
  12 ?            00:00:00 cpuhp/0
  13 ?            00:00:00 kdevtmpfs
  14 ?            00:00:00 netns
  15 ?            00:00:00 rcu_tasks_kthre
  16 ?            00:00:00 kauditd
  17 ?            00:00:00 khungtaskd
  18 ?            00:00:00 ...
```

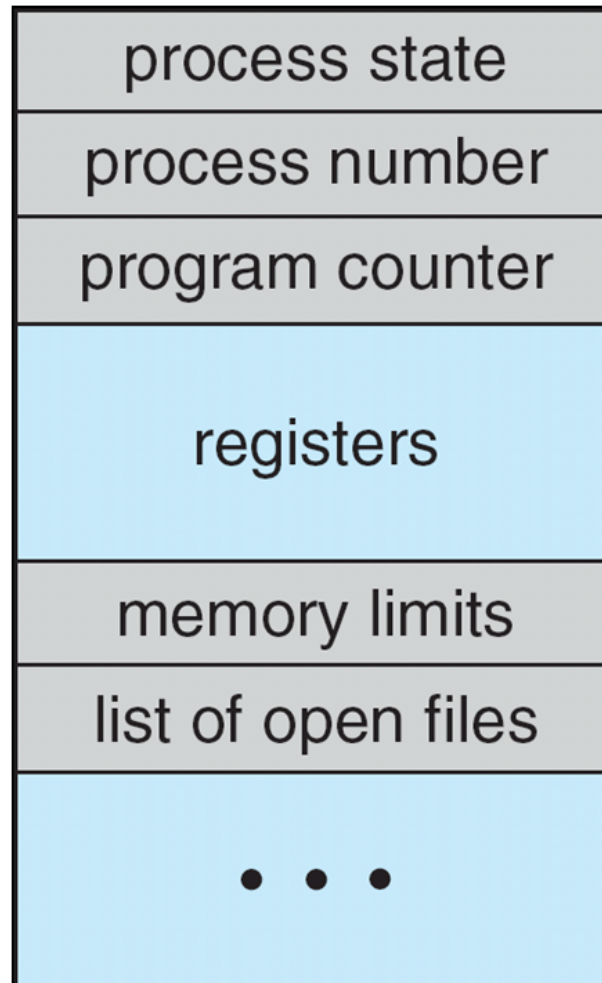
Windows (Task Manager)



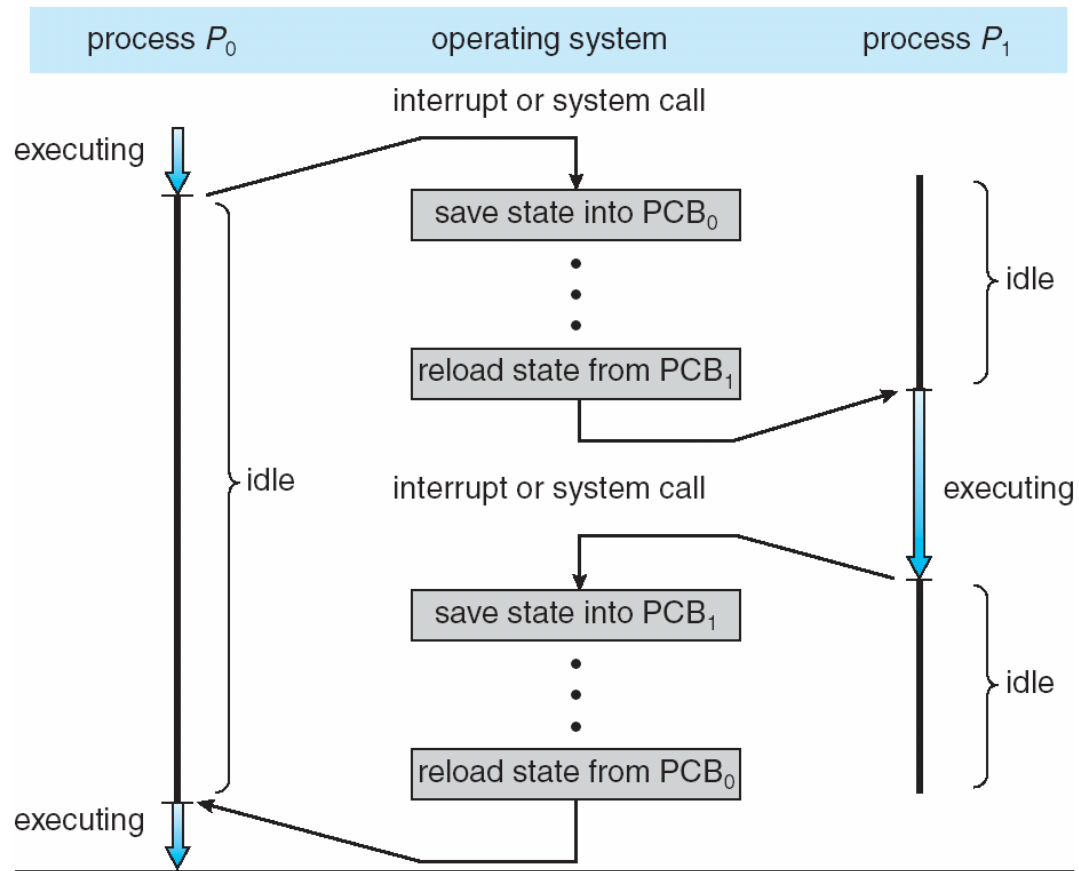
The screenshot shows the Windows Task Manager window with the 'Performance' tab selected. The window title is 'Task Manager'. The menu bar includes 'File', 'Options', and 'View'. Below the menu bar are tabs for 'Processes', 'Performance', 'App history', 'Startup', 'Users', 'Details', and 'Services'. The 'Performance' tab displays a table of system resource usage. The table has columns for 'Name', 'Status', 'CPU', 'Memory', 'Disk', and 'Network'. The 'CPU' column shows 100%, 'Memory' shows 55%, 'Disk' shows 7%, and 'Network' shows 0%. Below the table, there is a section for 'Apps (16)' listing various applications with their respective icons and resource usage. At the bottom, there is a 'Fewer details' button and an 'End task' button.

Name	Status	100% CPU	55% Memory	7% Disk	0% Network
Apps (16)					
Ashampoo ImageFX		0%	0.9 MB	0 MB/s	0 Mbps
Cookbook By Bewise		0%	16.6 MB	0 MB/s	0 Mbps
Dropbox		0%	15.8 MB	0.1 MB/s	0 Mbps
Grantophone		0%	5.1 MB	0 MB/s	0 Mbps
Internet Explorer		5.8%	2.3 MB	0.1 MB/s	0 Mbps
Kobo		0%	1.1 MB	0 MB/s	0 Mbps
Mail		0%	4.6 MB	0 MB/s	0 Mbps
News Republic		0%	33.9 MB	0 MB/s	0 Mbps
Photos		0%	21.0 MB	0 MB/s	0 Mbps
Pirates Love Daisies		0%	62.1 MB	0 MB/s	0 Mbps
Resource and Performanc...		7.3%	13.0 MB	0 MB/s	0 Mbps
SlapDash Podcasts		0%	32.4 MB	0 MB/s	0 Mbps

Process Control Block (PCB)



Execução de processos

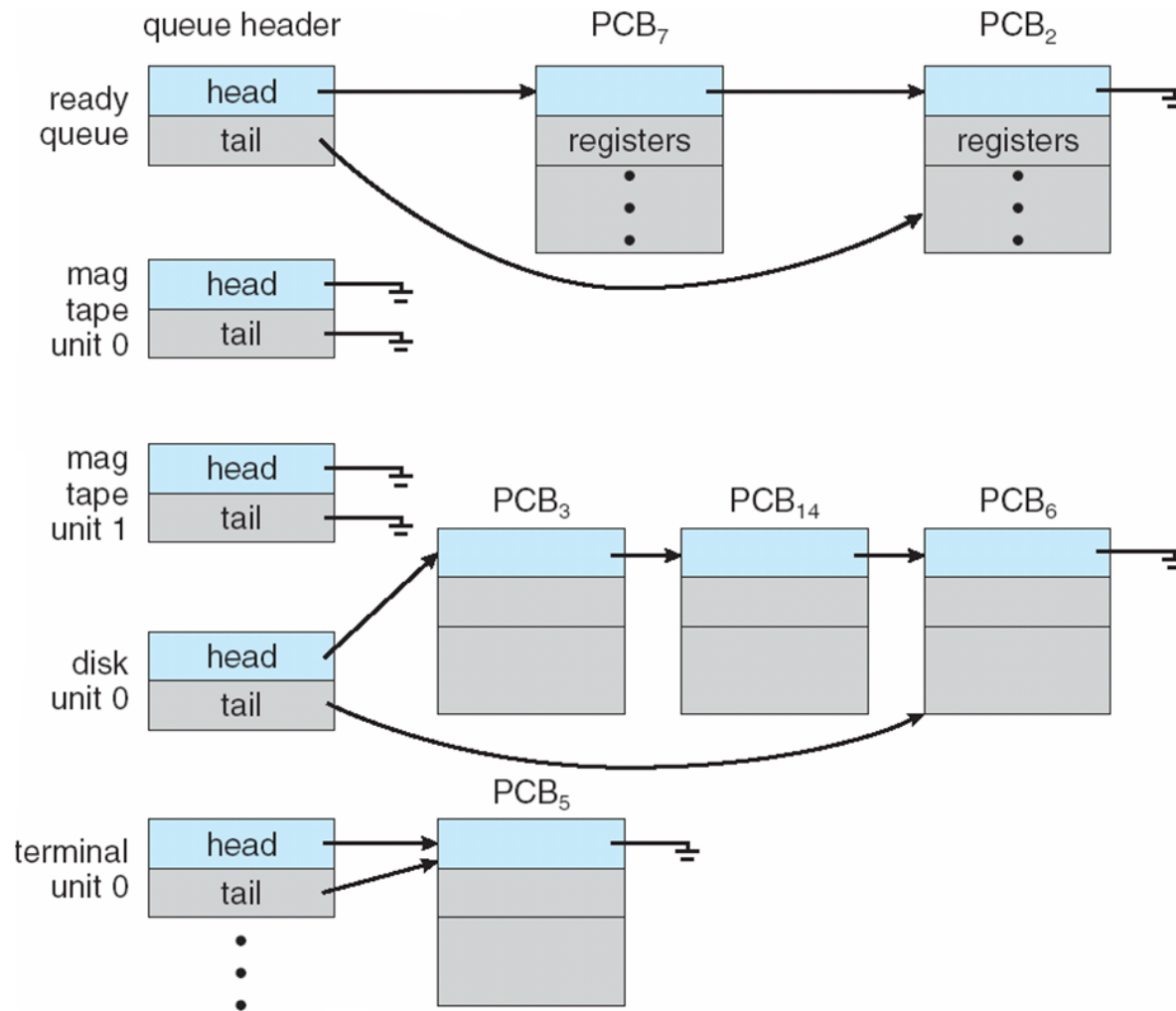


Escalonamento de processos

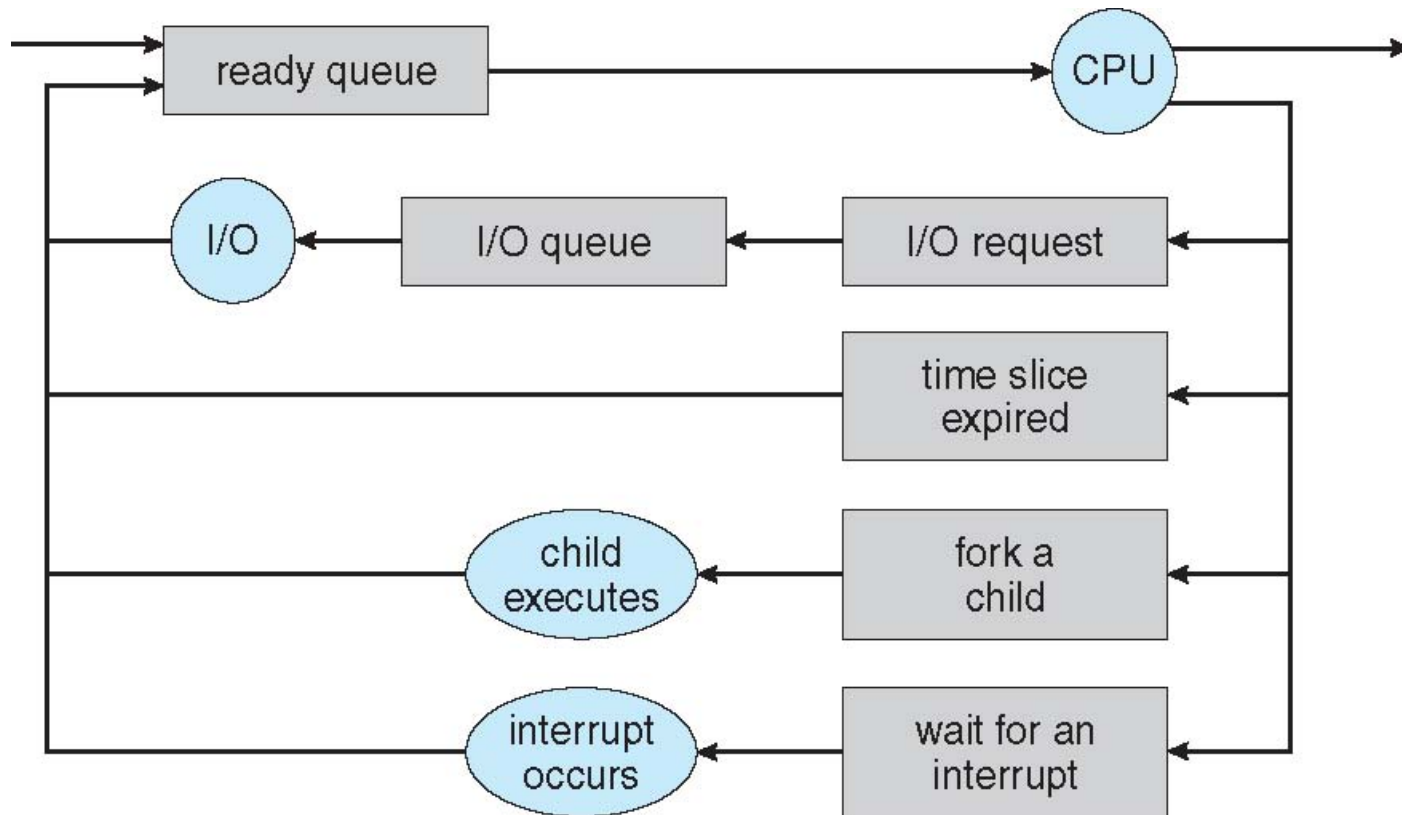
- Maximizar o uso do CPU
- Escalonador escolhe o próximo processo a ser executado
- Filas de espera de processos (queue)
 - **Job queue** - todos os processos no sistema
 - **Ready queue** - processos em memória prontos a ser executados
 - **Device queue** - processos a aguardar por dispositivos de I/O



Processos podem alternar entre as filas de espera..



Representação de escalonamento



Context switch

Guardar dados do processo anterior e ler dados do próximo processo...

- Demora tempo
- CPU não está a "produzir" trabalho útil
- Aumenta conforme o número de registos a ler/guardar

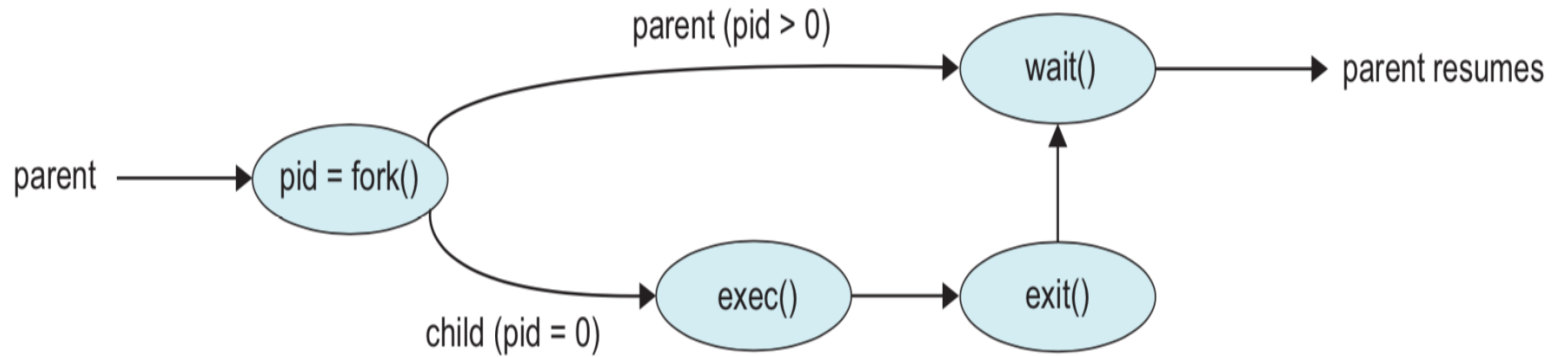
Criação de processos

Em Linux, usa-se a chamada de sistema `fork()` para criar processos..

```
int main()
{
    int pid = fork();

    if (pid == 0) {
        // Child process
    } else {
        // Parent process
    }
}
```

Fork...



Quantas vezes aparece "Hello World"?

```
int main()  
{  
    fork();  
    printf("Hello World");  
}
```

Quantas vezes aparece "Hello World"?

```
int main()  
{  
    fork();  
    fork();  
    printf("Hello World");  
}
```

Qual o valor das variáveis?

```
int main()
{
    int x = 1;
    int pid = fork();

    if (pid == 0) {
        // Child process
        x = x + 1;
        printf("%d", x);
    } else {
        // Parent process
        x = x - 1;
        printf("%d", x);
    }
}
```

Terminação de processos

- Utiliza-se a chamada de sistema *exit()*
- Processo retorna um inteiro para o pai (via *wait()*)

Exemplo com wait() e exit()

```
int main()
{
    if (fork() == 0) {
        // Child process
        printf("Hello from child process!");
        sleep(1);
        exit(0);
    } else {
        // Parent process
        int status;
        int pid = wait(&status);
        printf("Process %d has exited with status %d",
            pid, status);
    }
}
```

Interprocess communication (IPC)

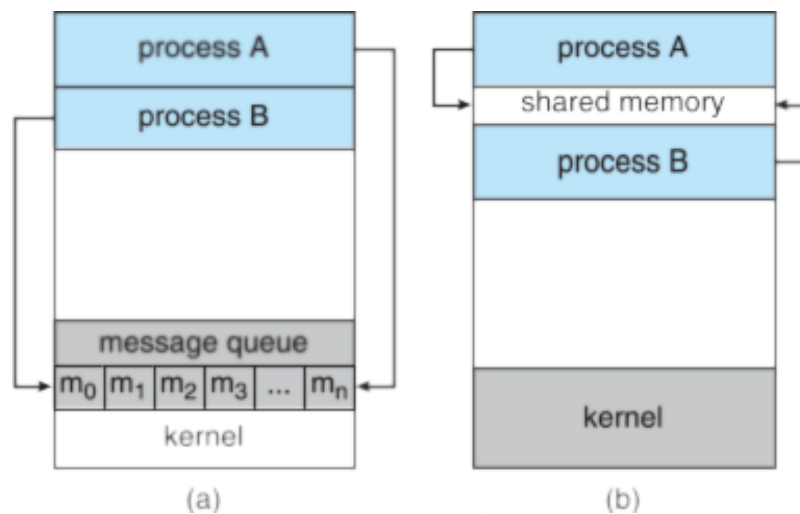
Processos podem ser:

- **Independentes** - não afectam nem são afectados por outros processos em execução
- **Cooperativos** - afectam ou são afectados por outros processos



Processos cooperativos necessitam partilhar informação!

Formas de IPC



a) Passagem de mensagens

b) Memória partilhada

Passagem de mensagens

send(P, message)

recv(P, message)

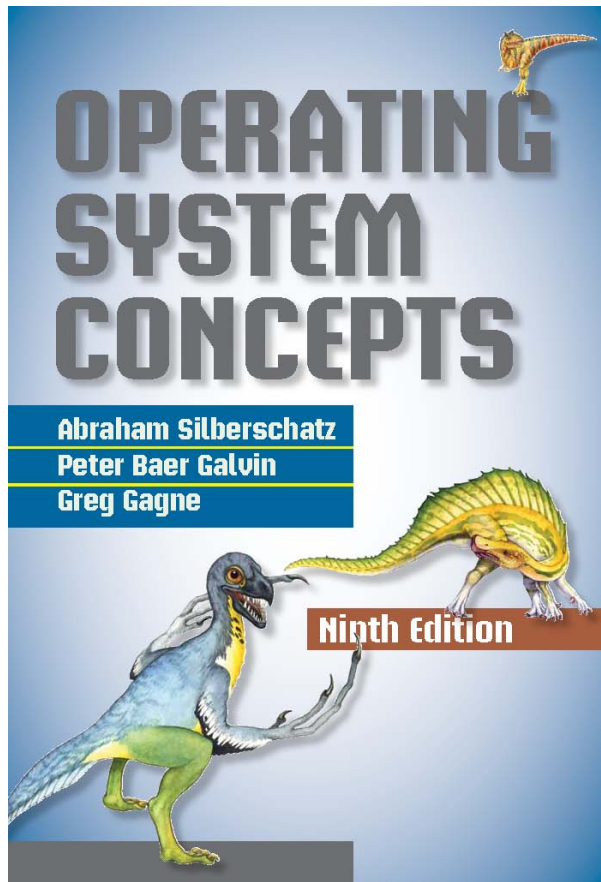
Memória partilhada - exemplo

```
int main() {  
    // Create shared memory map  
    int size = 64;  
    int protection = PROT_READ | PROT_WRITE;  
    int visibility = MAP_ANONYMOUS | MAP_SHARED;  
    void *shmem = mmap(NULL, size, protection,  
                        visibility, 0, 0);  
  
    if (fork() == 0) {  
        printf("Child process!\n");  
        char *msg = "Hello from child process..\n";  
        strcpy(shmem, msg);  
        exit(0);  
    } else {  
        printf("Parent process!\n");  
        sleep(1);  
        printf("shmem=%s", (char*) shmem);  
    }  
  
    return (EXIT_SUCCESS);  
}
```

Quiz...

Sumário

- Processo é um programa em execução
- É representado no SO por um Process Control Block
- Escalonador do SO escolhe próximo processo a executar
- Processos podem originar novos processos (ex: *fork()*)
- Processos cooperativos podem comunicar entre si (IPC)



Ler capítulo 3...