

Sistemas Operacionais: Processos



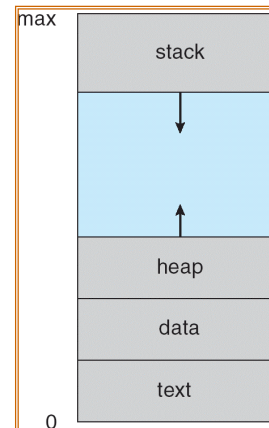
Processos

- Processos
- Escalonamento de processos
- Operações de processos
- Cooperação de processos
- Comunicação interprocessos



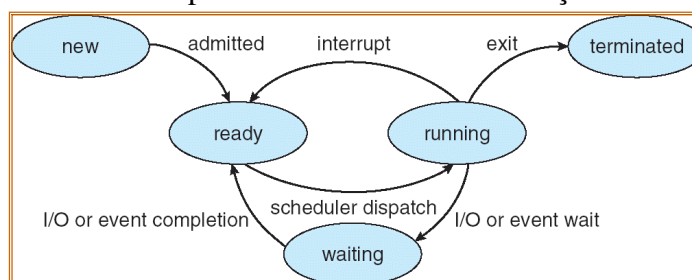
Processo

- “Unidade lógica de execução”
 - Em sistemas de lote: *job*
 - Em sistemas de tempo compartilhado: aplicações/programa ou *task*
- Processo: programa em execução
 - Contador de programa
 - Pilha
 - Seção de dados



Estados de um processo

- Estados de um processo durante sua execução
 - **Novo:** o processo acabou de ser criado
 - **Executando:** instruções sendo executadas
 - **Esperando:** esperando por algum evento
 - **Pronto:** esperando a CPU
 - **Finalizado:** o processo terminou sua execução

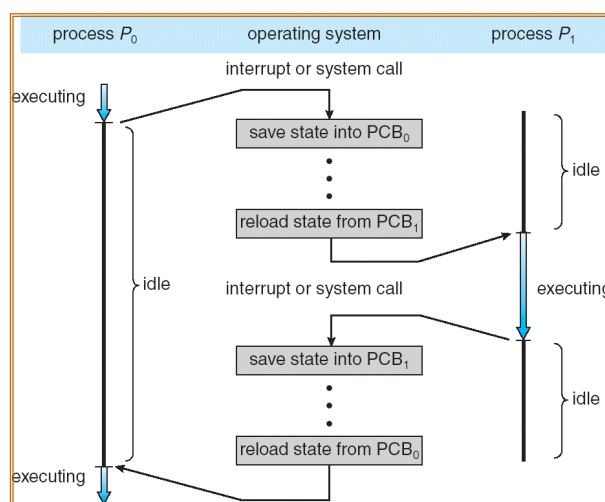


Estrutura de dados

- PCB: process control block, armazena informações sobre o processo
 - Process state
 - Program counter
 - CPU registers
 - CPU scheduling information
 - Memory-management information
 - Accounting information
 - I/O status information
- Veja: `/usr/src/linux/include/linux/sched.h`
 - `struct task_struct`



Troca de contexto



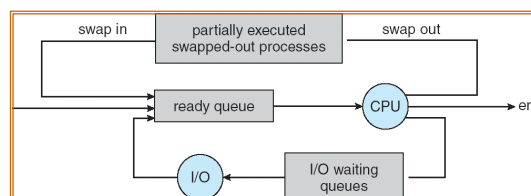
Filas de processos

- Filas encadeadas
 - Fila de processos prontos
 - Fila de processos esperando por dispositivos – E/S
- Processos migram nas diferentes filas



Escalonadores

- Escalonador intermediário
 - Coloca os processos aptos na fila de prontos
 - Execução freqüente => ms
- Escalonador de processos
 - Aloca o processador para um processo na fila de prontos
 - Execução não freqüente => segundos
- O escalonador de longo prazo controla o grau de multiprogramação (quantidade de processos prontos)



Tipos de processos

- Intensivo em CPU
 - Alta taxa de utilização de CPU
 - Pouca utilização de dispositivos de E/S
- Intensivo em E/S
 - Utilização da CPU em curtas rajadas
 - Alta taxa de acessos a dispositivos de E/S
 - Ex: Interativo



Troca de contexto

- Armazena as informações do processo em execução, e carrega o próximo processo a ganhar a CPU
- Dependente da arquitetura
- No Linux
 - Código: `switch_to` em `include/asm-i386/system.h`



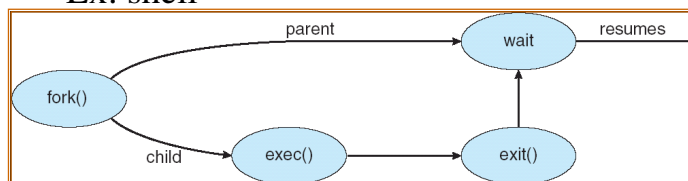
Criando processos

- O processo pai cria processos filhos, que criam outros processos, formando uma árvore de processos
- Estratégias de compartilhamento de recursos
 - O processo pai e filho compartilham todos os recursos
 - O filho compartilha um subconjunto de recursos do processo pai
 - Pai e filho não compartilham recursos
- Execução
 - Pai e filho executam concorrentemente
 - O processo pai espera até que o filho termine a execução
- Espaço de endereçamento
 - Duplica o espaço de endereçamento
 - Carrega um outro programa no espaço de endereçamento



UNIX

- Fork
 - Cria um processo filho que duplica o espaço de endereçamento
- Exec
 - Substitui o espaço no processo filho por um novo programa
 - Ex: shell

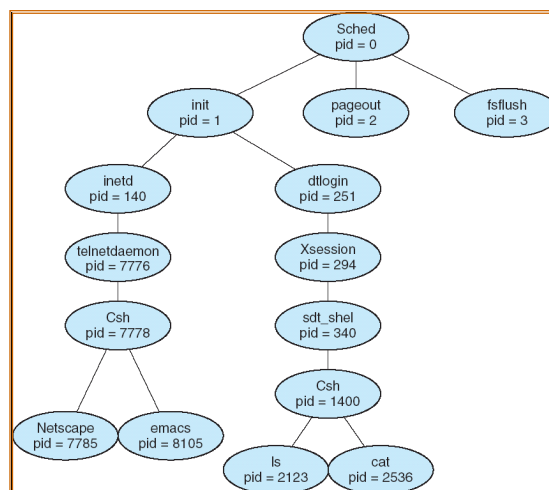


Fork

```
int main()
{
    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);
    }
}
```



Árvore de processos



Finalização de processos

- O processo executa até o final e solicita a finalização ao processo (**exit**)
 - Os dados de saída do filho são disponibilizados ao processo pai (via **wait**)
 - Os recursos do processo são desalocados
- O processo pai pode abortar a execução do processo filho (**abort**)
 - Razões
 - O processo filho excedeu o limite de recursos
 - A tarefa filha não é mais necessária
 - Finalização do processo pai
 - Alguns sistemas operacionais não deixam que os processos filhos continuem a execução
 - *Finalização em cascata*

