

Programação com Processos em Unix

Sistemas Operativos – DEI - IST

1



Processos em Unix

- Processos identificados por um PID *Process Identifier*
- O PID é um inteiro
- Alguns identificadores estão atribuídos a processos criados pelo sistema operativo
 - Processo 0 é o swapper (gestão de memória)
 - Processo 1 init inicialização do sistema

Sistemas Operativos – DEI - IST



Modelo de Segurança

- Cada utilizador no sistema identificado por User IDentifier (UID)
 - superuser (ou root) tem UID especial (zero)
- Para facilitar a partilha, o utilizador pertence a um ou mais grupos de utilizadores
 - Cada grupo identificado por um Group Identifier (GID)

Sistemas Operativos – DEI - IST

3



Autenticação de processos

- Cada processo corre em nome de um utilizador (UID)/grupo (GID)
- Inicialmente atribuídos ao primeiro processo criado quando o utilizador se autentica (log in)
 - Obtidos do ficheiro /etc/passwd no momento do login
- Processos filho criados a partir deste processo inicial, herdam UID/GID

Sistemas Operativos – DEI - IST



Autenticação de processos (II)

- Na verdade, há:
 - -Real UID e real GID
 - » Normalmente nunca mudam
 - Effective UID e effective GID
 - » Podem mudar temporariamente
 - E também há o saved UID/GID (fora da matéria)
- Quando o processo faz chamada sistema, o núcleo consulta o seu EUID/EGID para determinar se tem permissão

Sistemas Operativos – DEI - IST

6



Hierarquia de processos

- Processos relacionam-se de forma hierárquica
- Novo processo herda grande parte do contexto do processo pai
- Quando o processo pai termina os subprocessos continuam a executar-se
 - São adoptados pelo Processo de Inicialização (pid = 1)

Sistemas Operativos – DEI - IST



Criação de um Processo

id = fork(),

A função não tem parâmetros, em particular o ficheiro a executar.

Processo filho é uma cópia do pai:

- · O espaço de endereçamento é copiado
- Contexto de execução é copiado

A função retorna o PID do processo.

Este parâmetro assume valores diferentes consoante o processo em que se efetua o retorno:

- ao processo pai é devolvido o "pid" do filho
- ao processo filho é devolvido 0
- -1 em caso de erro

Sistemas Operativos – DEI - IST

11



Exemplo de fork

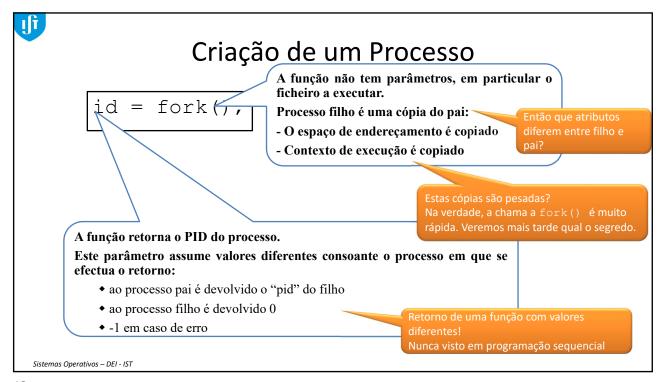
```
main() {
   int pid;
   pid = fork();
   if (pid == -1) /* tratar o erro */
   if (pid == 0) {

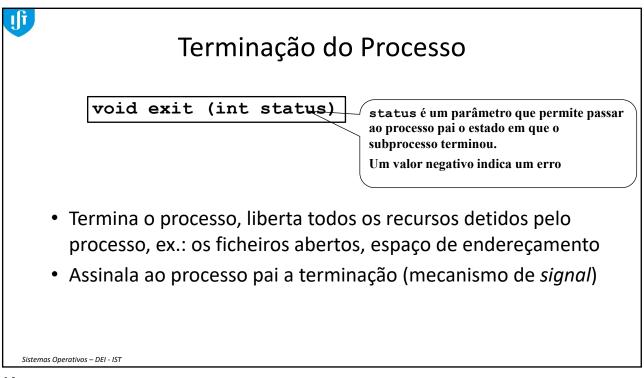
        /* código do processo filho */
   } else {

        /* código do processo pai */
   }

   /* instruções seguintes */
}
```

Sistemas Operativos – DEI - IST







exit()

- Até agora, a terminação de programa tem sido programada com return (int) na função main () do programa
- Qual a diferença?
- Nenhuma, pois o compilador assegura que um return () do main resulta em chamada a exit ()

```
main_aux(argc, argv) {
   int s = main(argc, argv);
   exit(s);
}
```

Função main () incluída pelo C

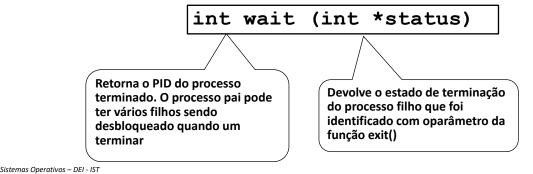
Sistemas Operativos – DEI - IST

15



Sincronização com a Terminação de um Subprocesso

- Em Unix existe uma função para o processo-pai se sincronizar com a terminação de um processo-filho
- Bloqueia o processo pai até que um dos filhos termine





Exemplo de Sincronização entre o Processo Pai e o Processo Filho

```
main () {
   int pid, estado;

pid = fork ();
   if (pid == 0) {
      /* algoritmo do processo filho */
      exit(0);
   } else {
      /* o processo pai bloqueia-se à espera da
            terminação do processo filho */
      pid = wait (&estado);
   }
}
```

Sistemas Operativos - DEI - IST

18



exit () não elimina todo o estado do processo

- São mantidos os atributos necessários para quando o pai chamar wait():
 - PID do processo terminado e do seu processo pai
 - Status da terminação
- Entre exit() e o wait() do processo pai, o processo diz-se estar no estado zombie
- Só depois de um wait () a informação sobre o processo é totalmente eliminada

Sistemas Operativos – DEI - IST



Conclusão

- O fork () permite criar um processo que é uma cópia do processo –pai
- No programa distingue-se o contexto em que se está a executar pelo retorno da função fork(): pid do filho no pai e 0 no filho
- Os processos tem associados um dono e um grupo definidos no login e herdados pelos processos filho
- A função wait () permite esperar pela terminação de um processo-filho

Sistemas Operativos – DEI - IST