Criação de Processos

Mário O. de Menezes
Faculdade de Computação e Informática – FCI
Universidade Presbiteriana Mackenzie

Identificação de Processos

Process ID

O Unix/Linux identifica os processos por um valor inteiro único chamado ID do processo.

Cada processo tem também um ID do processo pai, que é inicialmente o ID do processo que o criou. Se este processo pai termina, o processo é adotado por um processo do sistema de modo que o ID do processo pai sempre identifica um processo válido.

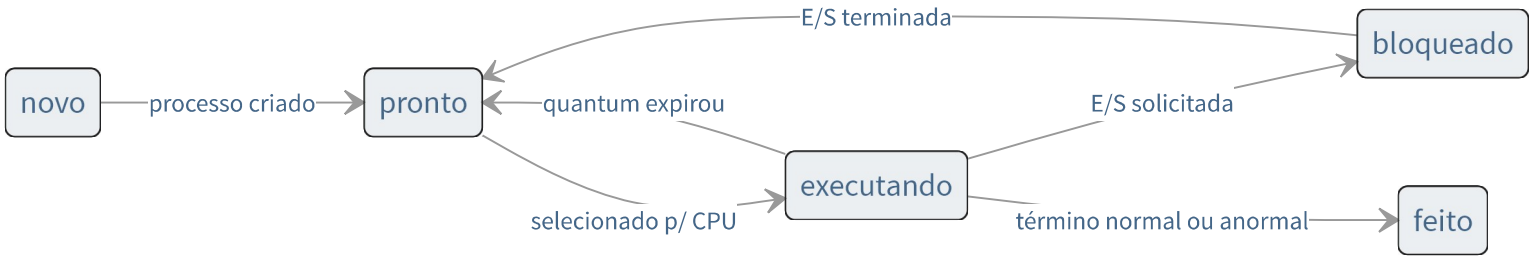
As funções `getpid` e `getppid` retornam o ID do processo e o ID do processo pai, respectivamente.

```c
#include <stdio.h>
#include <unistd.h>

int main (void) {
  printf("I am process %ld\n", (long)getpid());
  printf("My parent is %ld\n", (long)getppid());
  return 0;
}
```

Estados de um Processo

Estado	Significado
novo	Sendo criado
executando	Instruções estão sendo processadas
bloqueado	Esperando por um evento tal como E/S
pronto	Esperando para ser colocado em um processador
feito	Terminado

Diagrama de Estados de um Processo



Podemos examinar os processos em execução e visualizar seus estados utilizando o comando `ps -a`

Executando `ps -la` temos uma saída mais completa com várias informações adicionais sobre os processos.

```
$ps -la
F S   UID     PID    PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
0 S  1000   54212   49473  0  80   0 -  4342 do_pol pts/2    00:00:00 ssh
4 R  1000   56347    2795  0  80   0 -  3169 -      pts/1    00:00:00 ps
```

Campos reportados pelo comando ps

Campos reportados para as várias opções do comando ps na Extensão POSIX:XSI

cabeçalho	opção	significado
F	-l	flags (octal e aditivo) associados com o processo
S	-l	estado do processo
UID	-f,-l	ID de usuário do proprietário do processo
PID	(todos)	ID do processo
PPID	-f,-l	ID do processo pai
C	-f,-l	utilização de processador usada para escalonamento
PRI	-l	prioridade do processo
NI	-l	valor nice
ADDR	-l	endereço de memória do processo
SZ	-l	tamanho em blocos da imagem do processo
WCHAN	-l	evento no qual o processo está esperando
TTY	(todos)	terminal que está controlando
TIME	(todos)	tempo de execução cumulativo
CMD	(todos)	nome do comando (argumentos com a opção -f)

Criação de Processos

Um processo pode ser criado chamando-se `fork`

O processo chamador se torna o *pai* e o processo criado é chamado de *filho*

```c
#include <stdio.h>
#include <unistd.h>

int main(void) {
   int x;

   x = 0;
   fork();
   x = 1;
   printf("I am process %ld and my x is %d\n", (long)getpid(), x);
   return 0;
}
```

Desafio

Quantos *prints* aparecerão na tela, depois de executar o código acima?

`fork` – pai e filho

- Como se comporta o processo *filho* criado?
- É possível distinguir a execução do *pai* e do *filho*?

```c
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>

int main(void) {
   pid_t childpid;

   childpid = fork();
   if (childpid == -1) {
      perror("Failed to fork");
      return 1;
   }

   if (childpid == 0)      /* child code */
      printf("I am child %ld\n", (long)getpid());
   else                    /* parent code */
      printf("I am parent %ld\n", (long)getpid());
   return 0;
}
```

Importante!

A chamada de função `fork()` retorna **2** valores:

- o `PID` do processo *filho* criado **no processo pai**;
- `0` no **processo filho**.

Obtendo o Process ID

O que acontece quando o seguinte programa é executado?

Que valor será *printado* para a variável `mypid`?

```c
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>

int main(void) {
   pid_t childpid;
   pid_t mypid:

   mypid = getpid();
   childpid = fork();
   if (childpid == -1) {
      perror("Failed to fork");
      return 1;
   }

   if (childpid == )           /* child code */
     printf("I am child %ld, ID = %ld\n", (long)getpid(), (long)mypid);
   else                        /* child code */
     printf("I am parent %ld, ID = %ld\n", (long)getpid(), (long)mypid);
   return 0;
}
```

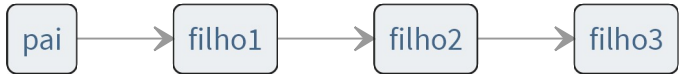
Criando uma cadeia de processos

Abaixo está o código de um programa que cria uma cadeia de *n* processos, onde *n* é um argumento de linha de comando.

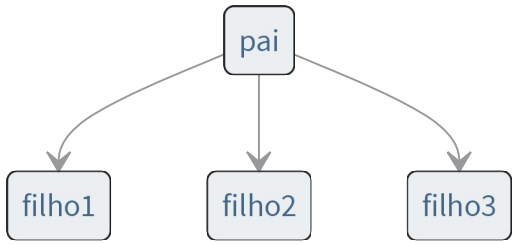
```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

int main (int argc, char *argv[]) {
   pid_t childpid = 0;
   int i, n;
   if (argc != 2){   /* check for valid number of command-line argumen
      fprintf(stderr, "Usage: %s processes\n", argv[0]);
      return 1;
   }
   n = atoi(argv[1]);
   for (i = 1; i < n; i++)
      if (childpid = fork())
         break;

   fprintf(stderr, "i:%d process ID:%ld parent ID:%ld child ID:%ld\n",
           i, (long)getpid(), (long)getppid(), (long)childpid);
   return 0;
}
```

Qual diagrama de criação de processos será criado pelo código ao lado?

Opção 1



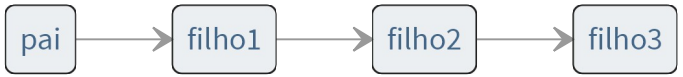
Opção 2



Exercícios

1. Rode o programa anterior para valores grandes de `n` (no máximo 40). As mensagens sempre estarão ordenadas pelo valor de `i`?
2. Usando um `n` grande ($15 < n <= 50$), o que acontece se o programa anterior escrevesse as mensagens para `sys.stdout`, usando `printf`, ao invés de para `sys.stderr`?
3. Altere a forma da cadeia de processos para ser como a Opção 2

Opção 1



Opção 2