

Sistemas Operacionais

Interação entre tarefas - mecanismos de comunicação

Prof. Carlos Maziero

DInf UFPR, Curitiba PR

Fevereiro de 2019

Conteúdo

- 1 Pipes UNIX
- 2 Filas de mensagens
- 3 Memória compartilhada

Pipes

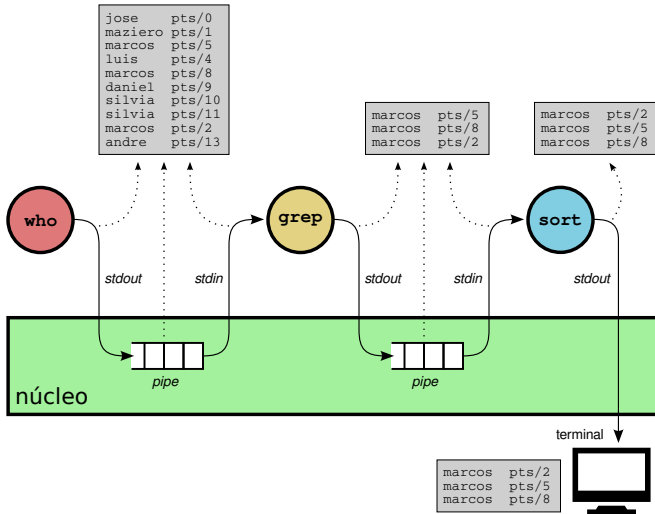
Mecanismo básico em sistemas Unix

Liga a saída de um processo à entrada de outro:

- `stdin`: entrada padrão (`scanf`, `getchar`, ...)
- `stdout`: saída padrão (`printf`, ...)
- `stderr`: saída de erro (`perror`, ...)

```
who | grep marcos | sort > login-marcos.txt
```

Pipes



Pipes nomeados (FIFOs)

```
1 # cria um pipe nomeado, cujo nome é/tmp/pipe
2 $ mkfifo /tmp/pipe
3
4 # mostra o nome do pipe no diretório
5 $ ls -l /tmp/pipe
6 prw-rw-r-- 1 mazierno mazierno 0 sept. 6 18:14 pipe|
7
8 # envia dados (saída do comando date) para o pipe nomeado
9 $ date > /tmp/pipe
10
11 # EM OUTRO TERMINAL, recebe dados do pipe nomeado
12 $ cat < /tmp/pipe
13 Thu Sep 6 2018, 18:01:50 (UTC+0200)
14
15 # remove o pipe nomeado
16 $ rm /tmp/pipe
```

Filas POSIX

Implementam o conceito de *mailbox*:

- **mq_open**: abre ou cria uma fila
- **mq_setattr**: ajusta atributos da fila (tamanho, etc)
- **mq_send**: envia uma mensagem para a fila; pode ser bloqueante ou não
- **mq_receive**: recebe uma mensagem da fila, idem
- **mq_timedsend**: versão semi-bloqueante
- **mq_timedreceive**: versão semi-bloqueante
- **mq_close**: fecha o descritor da fila
- **mq_unlink**: remove a fila do sistema

Filas POSIX - receptor

```

1  #define QUEUE "/my_queue"
2
3  mqd_t queue;           // descritor da fila de mensagens
4  struct mq_attr attr;  // atributos da fila de mensagens
5  int  msg ;            // as mensagens são números inteiros
6
7  // abre ou cria a fila com permissões 0666
8  if ((queue = mq_open (QUEUE, O_RDWR|O_CREAT, 0666, &attr)) < 0)
9  {
10     perror ("mq_open");
11     exit (1);
12 }
13
14 // recebe cada mensagem e imprime seu conteúdo
15 for (;;)
16 {
17     if ((mq_receive (queue, (void*) &msg, sizeof(msg), 0)) < 0)
18         perror("mq_receive:");
19     else
20         printf ("Received msg value %d\n", msg);
21 }
  
```

Filas POSIX - emissor

```

1  #define QUEUE "/my_queue"
2
3  // abre a fila de mensagens, se existir
4  if((queue = mq_open (QUEUE, O_RDWR)) < 0)
5  {
6      perror ("mq_open");
7      exit (1);
8  }
9
10 for (;;) {
11     msg = random() % 100 ; // valor entre 0 e 99
12
13     // envia a mensagem
14     if (mq_send (queue, (void*) &msg, sizeof(msg), 0) < 0)
15     {
16         perror ("mq_send");
17         exit (1);
18     }
19     printf ("Sent message with value %d\n", msg);
20     sleep (1) ;
21 }
  
```

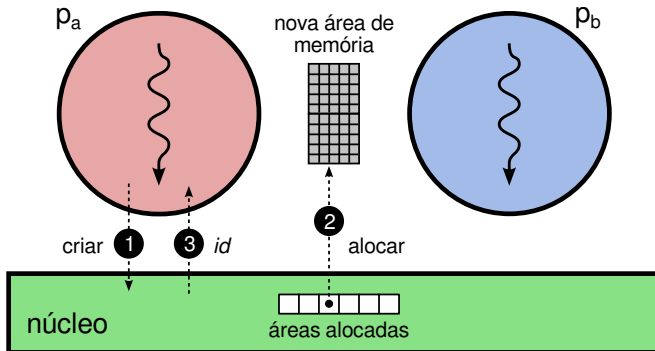

Memória compartilhada

Processos leem e escrevem na mesma área de memória:

- Normalmente proibido pelos mecanismos de hardware
- Núcleo ajusta mapas de memória para criar área compartilhada
- Comunicação rápida, sem interferência do núcleo
- Adequada para dados compartilhados
- Ausência de mecanismos de coordenação

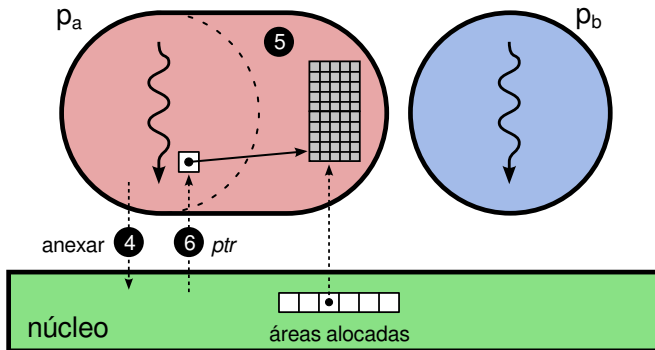
Memória compartilhada

- 1 p_a solicita ao núcleo uma área de memória compartilhada
- 2 o núcleo aloca uma nova área de memória
- 3 o núcleo devolve ao processo p_a o *id* dessa área



Memória compartilhada

- 4 p_a pede ao núcleo para anexar a área id ao seu espaço de endereçamento
- 5 o núcleo ajusta os mapas de memória de p_a
- 6 o núcleo devolve a p_a um ponteiro para a área id



Memória compartilhada

- 7 p_b executa os passos 4-6 e também recebe um ponteiro para a área id
- 8 Os processos p_a e p_b acessam a área compartilhada id

