Comunicação entre Processos com Pipes

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

Pipes

Pipe é uma forma de comunicação entre dois ou mais processos relacionados ou interrelacionados.

Pode ser dentro de um mesmo processo ou uma comunicação entre um processo *filho* e um *pai*.

Também pode ser multinível, tal como a comunicação entre o *pai*, o *filho* e o *neto*, etc.

A comunicação é realizada com um processo escrevendo no **pipe** e o outro lendo do **pipe**.

Para efetivar a comunicação com **pipe** utilizamos a chamada de sistemas `pipe()`, criando dois arquivos, um para escrita no arquivo e outro para leitura do arquivo.

A chamada de sistema `pipe()` tem a seguinte síntaxe:

```
        #include<unistd.h>

        int pipe(int pipedes[2]);
```

Usando pipes

A chamada de sistema acima vai criar um *pipe* para comunicação em uma direção, isto é, cria dois descritores, o primeiro é conectado para se ler do *pipe* e o outro é conectado para se escrever no *pipe*:

- `pipedes[0]`: para leitura;
- `pipedes[1]`: para escrita

Para as operações de leitura e escrita do *pipe*, são utilizadas as funções `open`, `read`, `write` e `close`

A seguir vamos analisar um programa exemplo para escrever e ler duas mensagens utilizando um *pipe*.

Algoritmo:

1. Cria um *pipe*.
2. Envia uma mensagem ao *pipe*.
3. Recupera (lê) a mensagem do *pipe* e escreve-a na saída padrão.
4. Envia outra mensagem ao *pipe*.
5. Recupera (lê) a mensagem do *pipe* e escreve-a na saída padrão.

Nota: a recuperação das mensagem pode ser feita após se enviar todas as mensagens.

Exemplo 1 - `simplepipe.c`

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

int main() {
   int pipefds[2];
   int returnstatus;
   char writemessages[2][20]={"Hi", "Hello"};
   char readmessage[20];
   returnstatus = pipe(pipefds);

   if (returnstatus == -1) {
      printf("Unable to create pipe\n");
      return 1;
   }

   printf("Writing to pipe - Message 1 is %s\n", writemessages[0]);
   write(pipefds[1], writemessages[0], sizeof(writemessages[0]));
   read(pipefds[0], readmessage, sizeof(readmessage));
   printf("Reading from pipe - Message 1 is %s\n", readmessage);
   printf("Writing to pipe - Message 2 is %s\n", writemessages[0]);
   write(pipefds[1], writemessages[1], sizeof(writemessages[0]));
   read(pipefds[0], readmessage, sizeof(readmessage));
   printf("Reading from pipe - Message 2 is %s\n", readmessage);
   return 0;
}
```

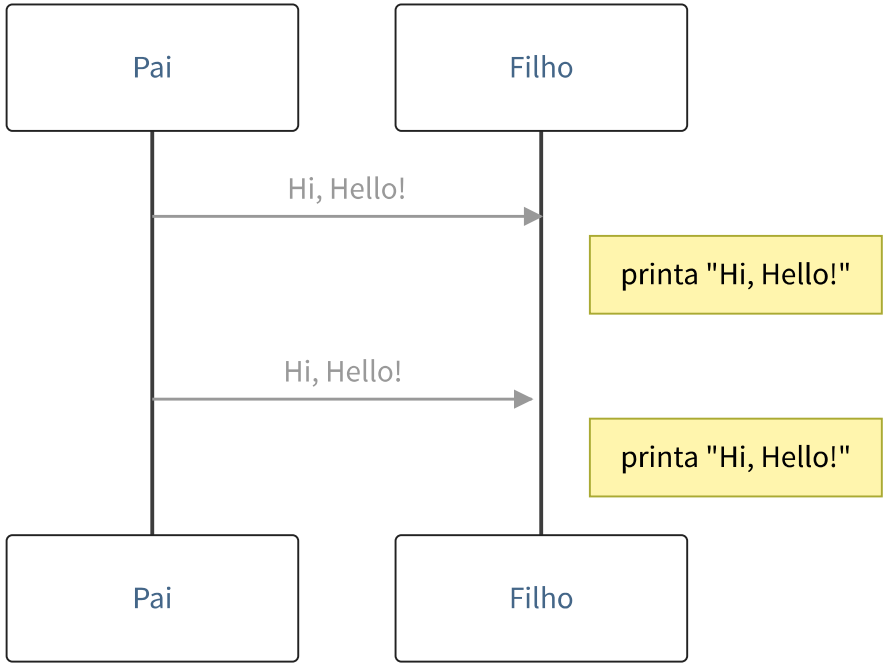
Idealmente devemos checar o resultado de *cada* chamada de sistema. Para simplificar, não estamos realizando esta checagem.

Exemplo 2

Neste exemplo vamos escrever e ler duas mensagens através do `pipe` usando os processos *pai* e *filho*.

Algoritmo | Código | Código – (cont.)

1. Cria o *pipe*.
2. Cria o processo *filho*.
3. Processo *pai* escreve no *pipe*.
4. Processo *filho* recupera a mensagem do *pipe* e escreve na saída padrão.
5. Repita os passos 3 e 4 mais uma vez.



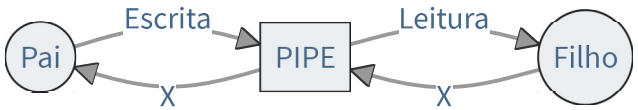
Comunicação nos dois sentidos

Para estabelecermos uma comunicação nos dois sentidos entre o processo *pai* e o processo *filho* criamos dois *pipes*.

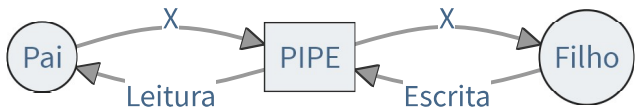
As etapas para isso são:

1. Cria dois *pipes*. O primeiro será utilizado para o processo *pai* **escrever** e o *filho* **ler**, vamos chamá-lo de `pipe1`. O segundo é para o processo *filho* **escrever** e o *pai* **ler**, vamos chamá-lo de `pipe2`.
2. Cria o processo *filho*.
3. Fecha nos as *pontas* desnecessárias, já que somente uma é necessária para comunicação.
    1. Fecha no processo *pai* a *ponta* de leitura do `pipe1` e a *ponta* de escrita do `pipe2`.
    2. Fecha no processo *filho* a *ponta* de escrita do `pipe1` e a *ponta* de leitura do `pipe2`.
4. Realiza a comunicação, como pretendido.

Pipe 1



Pipe 2



Exemplo 3

Vamos ver primeiramente o algoritmo para estabelecermos a comunicação nos dois sentidos.

Algoritmo | `twowayspipe.c` | `twowayspipe.c` (cont.)

1. Cria o `pipe1` para o procesos *pai* escrever e o *filho* ler.
2. Cria o `pipe2` para o processo *filho* escrever e o *pai* ler.
3. Fecha as pontas não necessárias dos `pipes` em ambos os lados dos processos *pai* e *filho*.
4. O processo *pai* escreve uma mensagem e o processo *filho* lê e a mostra na tela.
5. O processo *filho* escreve uma mensagem e processo *pai* a lê e mostra na tela.

Exercício com Pipes

1. Escreva um programa que utilize um processo *pai* e um processo *filho* para inverter uma palavra e também para verificar se ela é palíndrome ou não.

   O processo *pai* vai receber a palavra do usuário e vai escrevê-la no `pipe1`. O processo *filho* vai ler esta palavra, invertê-la, e escrever o resultado no `pipe2`. O processo *pai* então lê o resultado e imprime na tela.

   O processo *fiho* vai então escrever no `pipe3` o resultado da verificação se a palavra é palíndrome ou não.