

Unnamed Pipes

Comunicação entre processos

José Pedro Oliveira
(jpo@di.uminho.pt)

Grupo de Sistemas Distribuídos
Departamento de Informática
Escola de Engenharia
Universidade do Minho

Sistemas Operativos I
2006-2007



Conteúdo

- 1 Comunicação entre processos
 - Unnamed Pipes
 - Chamada ao sistema
- 2 Exercícios
- 3 Referências



José Pedro Oliveira Unnamed Pipes

Comunicação entre processos Unnamed Pipes

Pipes

Pipes

Pipes são a forma mais antiga de comunicação entre processos em sistemas UNIX.

Limitações

- 1 historicamente as pipes são *half-duplex*, isto é, a informação só flui numa direcção.
- 2 pipes só podem ser usadas entre processos que tenham um processo ancestral comum. Normalmente a pipe é criada por um processo, que em seguida invoca a chamada ao sistema `fork`, e em que a pipe é usada entre processo pai e filho.



José Pedro Oliveira Unnamed Pipes

Comunicação entre processos Chamada ao sistema

Exemplo de criação de uma pipe

Extracto de código

```

1  #include <unistd.h>
2
3  // ...
4
5  int fd[2];
6
7  if (pipe(fd) == -1) {
8      perror("pipe");
9      exit(1);
10 }
11
12 // ...

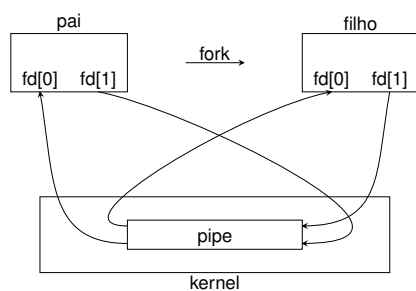
```



José Pedro Oliveira Unnamed Pipes

Comunicação entre processos Chamada ao sistema

Pipe half-duplex depois de um fork



José Pedro Oliveira Unnamed Pipes

José Pedro Oliveira Unnamed Pipes

Comunicação entre processos Chamada ao sistema

Chamada ao sistema pipe

Synopsis

```

#include <unistd.h>

int pipe(int filedes[2]);

```

Valores de retorno

-1 - erro
0 - ok

Descritores de ficheiros

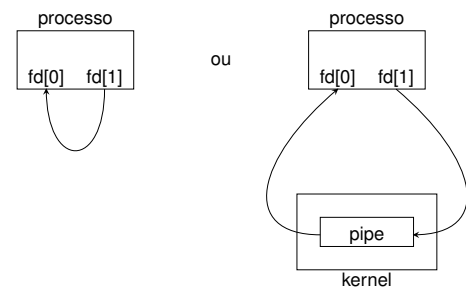
<code>filedes[0]</code>	aberto em modo de leitura
<code>filedes[1]</code>	aberto em modo de escrita



José Pedro Oliveira Unnamed Pipes

Comunicação entre processos Chamada ao sistema

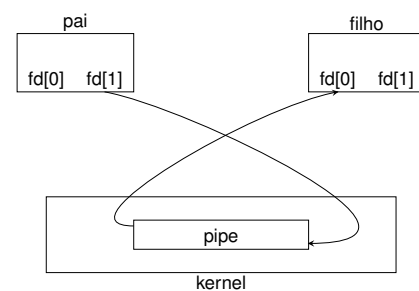
Pipe half-duplex



José Pedro Oliveira Unnamed Pipes

Comunicação entre processos Chamada ao sistema

Pipe half-duplex: canal de comunicação pai -> filho



José Pedro Oliveira Unnamed Pipes

Pipe half-duplex: canal de comunicação pai -> filho

```

1  int pai2filho [2];
2
3  if (pipe(pai2filho) == -1) { /* erro */ }
4
5  switch(fork()) {
6      case -1:
7          perror("fork"); exit(2);
8      case 0:
9          close(pai2filho[1]);
10         // write(pai2filho[0], ..., ...)
11         break;
12     default:
13         // pai */
14         close(pai2filho[0]);
15         // write(pai2filho[1], ..., ...)
16         wait(NULL);
17         break;
18     }

```

José Pedro Oliveira

Unnamed Pipes

Comunicação entre processos

Chamada ao sistema

pipe

Um dos extremos da pipe está fechado

Quando um dos extremos da pipe é fechado, aplicam-se as seguintes regras:

- quando se tenta ler de uma pipe cuja extremidade de escrita já se encontra fechada, a chamada ao sistema **read** retorna 0 (zero) para indicar o fim do ficheiro.
- quando se tenta escrever numa pipe cuja extremidade de leitura já se encontra fechada, é gerado o sinal **SIGPIPE**. Se o sinal for ignorado ou apanhado, a chamada ao sistema **write** retorna -1 com **errno** igual a **EPIPE**.



José Pedro Oliveira

Unnamed Pipes

Comunicação entre processos

Chamada ao sistema

Sinal SIGPIPE

SIGPIPE

```

1  // includes: stdio.h, stdlib.h, unistd.h, signal.h
2
3  int main(void)
4  {
5      int fd[2];
6
7      signal(SIGPIPE, SIG_IGN);
8
9      if (pipe(fd) == -1) { perror("pipe"); exit(1); }
10
11     close(fd[0]);
12     if (write(fd[1], "Teste\n", 6) == -1) { perror("write"); }
13     close(fd[1]);
14
15     write(STDOUT_FILENO, "Fim\n", 4);
16
17     return 0;
18 }

```

José Pedro Oliveira

Unnamed Pipes

Comunicação entre processos

Chamada ao sistema

Comunicação via *unnamed pipes*

Implementar

- cat /etc/passwd | grep bash
- cat /etc/passwd | grep bash | wc -l
- rpm -qa | grep ^vim | xargs rpm -q | grep /bin | wc -l



José Pedro Oliveira

Unnamed Pipes

Pipe half-duplex: canal de comunicação pai -> filho

```

1  int pai2filho [2], i;
2
3  if (pipe(pai2filho) == -1) { /* erro */ }
4
5  switch(fork()) {
6      case -1:
7          perror("fork"); exit(2);
8      case 0:
9          close(pai2filho[1]);
10         read(pai2filho[0], &i, sizeof(int));
11         break;
12     default:
13         // pai */
14         close(pai2filho[0]);
15         i = 1234;
16         write(pai2filho[1], &i, sizeof(int));
17         wait(NULL);
18         break;
19     }

```

José Pedro Oliveira

Unnamed Pipes

Comunicação entre processos

Chamada ao sistema

Sinal SIGPIPE

SIGPIPE

```

1  // includes: stdio.h, stdlib.h, unistd.h
2
3  int main(void)
4  {
5      int fd[2];
6
7      if (pipe(fd) == -1) {
8          perror("pipe"); exit(1);
9      }
10
11     close(fd[0]);
12     write(fd[1], "Teste\n", 6);
13     close(fd[1]);
14
15     write(STDOUT_FILENO, "Fim\n", 4);
16
17     return 0;
18 }

```

José Pedro Oliveira

Unnamed Pipes

Comunicação entre processos

Chamada ao sistema

Comunicação via *unnamed pipes*

Passos

- criar as pipes necessárias
- gerar o(s) processo(s) filho
- fechar/duplicar descritores de ficheiros para associar correctamente os extremos das pipes
- fechar os extremos não necessários
- realizar as actividades de comunicação
- fechar restantes descritores de ficheiros
- se necessário, esperar que os processos filhos terminem



José Pedro Oliveira

Unnamed Pipes

Comunicação entre processos

Chamada ao sistema

Exemplo: cat /etc/passwd | grep bash

```

1  int fd[2];
2
3  if (pipe(fd) == -1) { perror("pipe"); exit(1); }
4
5  switch(fork()) {
6      case -1:
7          perror("fork"); exit(2);
8      case 0:
9          close(fd[0]);
10         dup2(fd[1], STDOUT_FILENO);
11         close(fd[1]);
12         execlp("cat", "cat", "/etc/passwd", NULL);
13         exit(3);
14     default:
15         // pai */
16         close(fd[1]);
17         dup2(fd[0], STDIN_FILENO);
18         close(fd[0]);
19         execlp("grep", "grep", "bash", NULL);
20         exit(4);
21     }

```

José Pedro Oliveira

Unnamed Pipes

Exemplo: cat /etc/passwd | grep bash

```

1  if (pipe(fd) == -1) { perror("pipe"); exit(1); }
2
3  for (i=0; i<2; i++) {
4      p = fork();
5      if (p == -1) {
6          perror("fork"); exit(2);
7      } else if (p == 0) {
8          switch(i) {
9              case 0:          /* primeiro filho */
10             // ...
11             exit(3);
12             case 1:          /* segundo filho */
13             // ...
14             exit(4);
15         }
16     }
17 }
18
19 close(fd[0]); close(fd[1]);
20 for (i=0; i<2; i++) { wait(NULL); }
```

José Pedro Oliveira

Unnamed Pipes

Exercícios

Conteúdo

- 1 Comunicação entre processos
 - Unnamed Pipes
 - Chamada so sistema
- 2 Exercícios
- 3 Referências



José Pedro Oliveira

Unnamed Pipes

Referências

Comunicação via unnamed pipes

Enunciados

- 1 Processo pai envia um número inteiro para o processo filho. Este multiplica o valor recebido por dois e devolve o resultado ao processo pai.
- 2 Processo pai envia dois números inteiros para o processo filho. Este adiciona-os e devolve o resultado ao processo pai.
- 3 Processo pai envia um array de n números inteiros para o processo filho. Este adiciona todos os elementos do array e devolve o resultado ao processo pai.
- 4 Processo pai envia uma string para o processo filho. Este converte todos os caracteres da string recebida para maiúsculas e devolve a string resultante ao processo pai.



José Pedro Oliveira

Unnamed Pipes

Referências

Conteúdo

- 1 Comunicação entre processos
 - Unnamed Pipes
 - Chamada so sistema
- 2 Exercícios
- 3 Referências



José Pedro Oliveira

Unnamed Pipes

Referências

Bibliografia

- **Advanced Programming in the UNIX Environment, 2nd ed.**
W. Richard Steven, Stephen A. Rago
<http://www.apuebook.com/>
 - Capítulo 15 - Interprocess Communication
- **Advanced UNIX Programming, 2nd ed.**
Marc J. Rochkind
<http://www.basepath.com/aup/>
 - Capítulo 6 - Basic Interprocess Communication
- **Linux Programming by Example: The Fundamentals**
Arnold Robbins
<http://authors.phptr.com/robbins/>
 - Capítulo 9 - Process Management and Pipes



José Pedro Oliveira

Unnamed Pipes