n prev

Next: Utilização dos tubos **Up:** Exemplos globais **Previous:** Implementação de um comando

Comunicação entre pais e filhos usando um tubo

Exemplo 1: Envio de uma mensagem ao usuário

Este programa permite que processos troquem mensagens com ajuda do sistema de correio eletrônico.

```
          /* arquivo test_pipe_mail.c */

/* teste do envio de um mail usando tubos */

#include <errno.h>
#include <signal.h>
#include <unistd.h>
#include <stdio.h>
#include <sys/wait.h>
#include <sys/types.h>

int main()
{
    FILE *fp;
    int pid, pipefds[2];
    char *username, *getlogin();
    /* da o nome do usuario */
    if ((username = getlogin()) == NULL) {
        fprintf(stderr, "quem e voce?\n");
        exit(1);    }
     /* Cria um tubo. Isto deve ser feito antes do fork para que
      * o filho possa herdar esse tubo
      */
    if (pipe(pipefds) < 0) { perror("Error pipe");
        exit(1);    }
    if ((pid = fork()) < 0) {    perror("Error fork");
        exit(1); }
     /* Codigo do filho:
      * executa o comando mail e entao envia ao username
      * a mensagem contida no tubo */
    if (pid == 0) {
  /* redirige a stdout para o tubo; o comando executado em seguida tera
   como entrada (uma mensagem) a leitura do tubo */
        close(0);
        dup(pipefds[0]);
        close(pipefds[0]);
  /* fecha o lado de escrita do tubo, para poder ver a saida na tela */
        close(pipefds[1]);
        /* executa o comando mail */
```

```
        execl("/bin/mail", "mail", username, 0);
        perror("Error execl");
        exit(1);
    }
    /* Codigo do pai:
     * escreve uma mensagem no tubo */
    close(pipefds[0]);
    fp = fdopen(pipefds[1], "w");
    fprintf(fp, "Hello from your program.\n");
    fclose(fp);
    /* Espera da morte do processo filho */
    while (wait((int *) 0) != pid) ;
    exit(0);
}
```

Resultado da execução: O usuário que executa o programa vai enviar a si mesmo um mensagem por correio eletrônico. A mensagem deve ser exatamente igual à mostrada a seguir:

```
Date: Fri, 13 Oct 2000 10:28:34 -0200
From: Celso Alberto Saibel Santos <saibel@leca.ufrn.br>
To: saibel@leca.ufrn.br

Hello from your program.
```

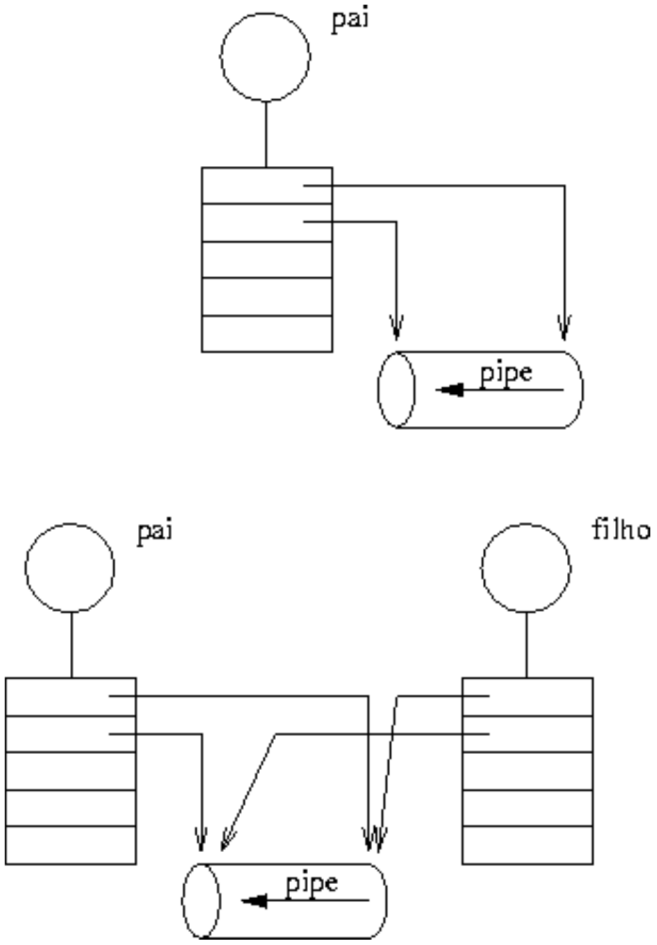

Figura 4.1: Compartilhamento de um tubo entre pai e filho

Exemplo 2: Enfoca mais uma vez a herança dos descritores através de um `fork()`. O programa cria um tubo para a comunicação entre um processo pai e seu filho.

```
                /* arquivo test_pipe_fork.c */

/* Testa heranca dos descritores na chamada do fork().
 * O programa cria um tubo e depois faz um fork. O filho
 * vai se comunicar com o pai atraves desse tubo
*/

#include <errno.h>
#include <stdio.h>
#include <unistd.h>

#define DATA "Testando envio de mensagem usando pipes"

int main()
{
   int sockets[2], child;
   char buf[1024];
```

```
   /* criacao de um tubo */
   if ( pipe(sockets) == -1 ) {
          perror("Error opening stream socket pair") ;
          exit(10);
   }
   /* criacao de um filho */
   if ( (child = fork()) == -1)
     perror ("Error fork") ;
   else if (child) {
     /* Esta ainda e a execução do pai. Ele lê a mensagem do filho */
     if ( close(sockets[1]) == -1) /* fecha o descritor nao utilizado */
          perror("Error close") ;
     if (read(sockets[0], buf, 1024) < 0 )
       perror("Error: reading message");
     printf("-->%s\n", buf);
     close(sockets[0]);
   } else {
     /* Esse e o filho. Ele escreve a mensagem para seu pai */
      if ( close(sockets[0]) == -1) /* fecha o descritor nao utilizado */
        perror("Error close") ;
      if (write(sockets[1], DATA, sizeof(DATA)) < 0 )
        perror("Error: writing message");
      close(sockets[1]);
   }
   sleep(1);
   exit(0);
}
```

Resultado da execução:

```
euler:~/> test_pipe_fork
-->Testando envio de mensagem usando pipes
euler:~/>
```

Um tubo é criado pelo processo pai, o qual logo após faz um `fork`. Quando um processo faz um `fork`, a tabela de descritores do pai é automaticamente copiada para o processo filho.

No programa, o pai faz um chamada de sistema `pipe()` para criar um tubo. Esta rotina cria um tubo e inclui na tabela de descritores do processos os descritores para os *sockets* associados às duas extremidades do tubo. Note que as extremidades do tubo não são equivalentes: o *sockets* com índice 0 está sendo aberto para leitura, enquanto que o de índice 1 está sendo aberto somente para a escrita. Isto corresponde ao fato de que a entrada padrão na tabela de descritores é associada ao primeiro descritor, enquanto a saída padrão é associada ao segundo.

Após ser criado, o tubo será compartilhado entre pai e filho após a chamada `fork`. A figura 4.5.1 ilustra o efeito da chamada `fork`.

A tabela de descritores do processo pai aponta para ambas as extremidades do

tubo. Após o `fork`, ambas as tabelas do pai e do filho estarão apontando para o mesmo tubo (herança de descritores). O filho então usa o tubo para enviar a mensagem para o pai.

---

n prev

Next: Utilização dos tubos **Up:** Exemplos globais **Previous:** Implementação de um comando

Celso Alberto Saibel Santos 2000-11-14