Laboratório de Pesquisa em Redes e Multimídia

Inter-process Communication (IPC)

Comunicação entre processos (2)

Filas (FIFOs, *Named Pipes*)

Universidade Federal do Espírito Santo
Departamento de Informática

Fila (FIFO, Named Pipe)

- Trata-se de uma extensão do conceito de *pipe*.
  - *Pipes* só podem ser usados por processos que tenham um ancestral comum.
  - Filas (FIFOs – First In First Out), também designados de "tubos nomeados" ("*named pipes*"), permitem a comunicação entre processos não relacionados.
- As Filas:
  - são referenciadas por um identificador dentro do sistema de arquivos
  - persistem além da vida do processo
  - são mantidas no sistema de arquivos até serem apagadas (ou seja, precisam ser eliminadas quando não tiverem mais uso).
- Normalmente são implementadas através de arquivos especiais (tipo: pipe).
  - Um processo abre a Fila para escrita, outro para leitura.

Criação de Filas (1)

- Uma fila é criada pela chamada de sistema:

```
POSIX: #include <sys/stat.h>
       int mkfifo(char *,mode_t);
```

  - 1º parâmetro: nome do arquivo.
  - 2º parâmetro: identifica as permissões de acesso, iguais a qualquer arquivo, determinados por OU de grupos de bits.
- As permissões de acesso também podem ser indicados por 3 dígitos octais, cada um representando os valores binários de rwx (Read,Write,eXecute).
  - Exemplo: modo 644 indica permissões de acesso:
    - Dono: 6 = 110 (leitura e escrita)
    - Grupo e Outros: 4 = 100 (leitura)

Criação de Filas (2)

- Uma fila também pode ser criada, via shell, por meio do comando:

```
#mkfifo [-m modo] fichID
```

- Exemplo 1:

```
[rgc@asterix]$ mkfifo -m 644 tubo
[rgc@asterix]$ ls -l tubo
prw-r--r-- 1 rgc docentes 0 2008-10-11 15:56 tubo
[rgc@asterix]$
```

OBS: **p** indica que "tubo" é um arquivo do tipo named pipe

- Exemplo 2:

```
#mkfifo teste
#cat < teste /* o pipe fica esperando até obter algum dado */
Em outra tela execute:
# ls > teste /* a saída do comando ls será redirecionada para o
pipe nomeado "teste" */
```

Eliminação de Filas

- Uma fila é eliminada pela seguinte chamada ao sistema:

```
POSIX:#include <unistd.h>
      int unlink(char *);
```

  - 1º parâmetro: nome do arquivo.

- Uma fila também é eliminada via shell, usando o comando:

```
#rm fichID
```

Abertura de Filas (1)

- Antes de ser usada, a fila tem de ser aberta pela chamada de sistema:

```
POSIX: #include <sys/types.h>
       #include <sys/stat.h>
       #include <fcntl.h>
       int open(char *,int);
```

  - 1º parâmetro: nome do arquivo.
  - 2º parâmetro : formado por bits que indicam:
    - Modos de acesso: `O_RDONLY` (leitura apenas) ou `O_WRONLY` (escrita apenas)
    - Opções de abertura: `O_CREAT` (criado se não existir)
    - `O_NONBLOCK` (operação de E/S não são bloqueadas)
  - O valor de retorno é o descritor da fila (positivo) ou erro (-1).

Abertura de Filas (2)

- Regras aplicadas na abertura de filas:
  - Se um processo tentar abrir uma fila em modo de leitura, e nesse instante não houver um processo que tenha aberto a fila em modo de acesso de escrita, o processo fica bloqueado, exceto se:
    - a opção `O_NONBLOCK` tiver sido indicada no momento da leitura (nesse caso, é devolvido o valor -1 e `errno` fica com valor `ENXIO`).
  - Se um processo tentar abrir uma fila em modo de escrita, e nesse instante não houver um processo que tenha aberto a fila em modo de acesso de leitura, o processo fica bloqueado, exceto se:
    - a opção `O_NONBLOCK` tiver sido indicada no momento da escrita (nesse caso, é devolvido o valor -1 e `errno` fica com valor `ENXIO`).

Leitura e Escrita em Filas (1)

- A comunicação em uma fila é feita pelas mesmas chamadas de sistema dos *pipes*:

```
POSIX: #include <unistd.h>
       ssize_t read(int, char *,int);
       ssize_t write(int, char *,int);
```

- Regras aplicadas aos processos escritores:
  - Escrita para uma fila que ainda não foi aberta para leitura gera o sinal `SIGPIPE` (ação por omissão de terminar o processo. Se ignorado `read` retorna -1 com `errno` igual a `EPIPE`).
  - Após o último processo escritor tiver encerrado a fila, os processos leitores recebem `EOF`.

Exemplo

- Dois processos *writer* enviam mensagens para o processo *reader* através de uma fila.
  - O identificador da fila e o comprimento da memória tampão é definida no arquivo à parte.

```
#define LEN 100
#define FNAME "testFIFO"
```

Exemplo (cont.)

Writer.C

```
#include <stdio.h>
#include <string.h>
#include <sys/file.h>

#include "defs.h"

main() {
   int fd, i;
   char msg[LEN];
   do {
      fd=open(FNAME,O_WRONLY);
      if (fd==-1) sleep(1); }
   while (fd==-1);
   for( i=1;i<=3;i++ ) {
      sprintf(msg,"Hello no %d from process %d\n",i,getpid());
      write( fd,msg,strlen(msg)+1 );
      sleep(3); }
      close(fd); }
```

Exemplo (cont.)

Reader.C

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/file.h>
#include "defs.h"

int readChar(int fd, char *buf) {
   int n;
   do n=read(fd,buf,1);
   while (n>0 && *buf++!='\0');
   return n>0; }

main() {
   int fd;
   char str[LEN];
   mkfifo(FNAME,0660);
   fd=open(FNAME,O_RDONLY);
   if (fd<0) { printf("Erro na abertura da fila\n"); exit(1); }
   while (readChar(fd,str)) printf("%s",str);
   close(fd); }
```

Exemplo (cont.)

```
[rgc@asterix FIFO]$ reader & writer & writer &
[1] 7528
[2] 7529  ⟺ PIDs dos processos lançados
[3] 7530
[rgc@asterix FIFO]$ Hello no 1 from process 7530
Hello no 1 from process 7529
Hello no 2 from process 7530
Hello no 2 from process 7529
Hello no 3 from process 7530
Hello no 3 from process 7529

[1]  Done                    reader
[2]- Done                    writer
[3]+ Done                    writer
[rgc@asterix FIFO]$
```

Lançados 1 leitor e 2 escritores

Exemplo (cont.)

```
[rgc@asterix FIFO]$ ls -l
total 48
-rw-r----- 1 rgc ec-ps          42   2007-05-17 15:17 defs.h
-rwxr----- 1 rgc ec-ps        5420   2007-05-17 15:45 reader
-rw-r--r-- 1 rgc ec-ps         442   2007-05-17 15:45 reader.c
prw-r----- 1 rgc docentes        0   2008-10-11 16:01 testFIFO
-rwxr----- 1 rgc ec-ps        5456   2007-05-17 15:23 writer
-rw-r--r-- 1 rgc ec-ps         371   2007-05-17 15:23 writer.c

[rgc@asterix FIFO]$ rm testFIFO
rm: remove fifo `testFIFO'? y
[rgc@asterix FIFO]$
```

Observe que a fila não havia sido eliminada pelos programas (arquivo testFIFO tem tipo **p**, de **named pipe**).