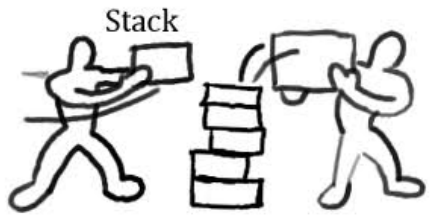
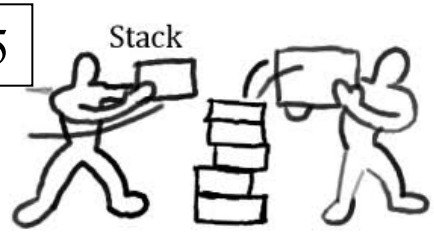


Programação de Sistemas

FIFOs

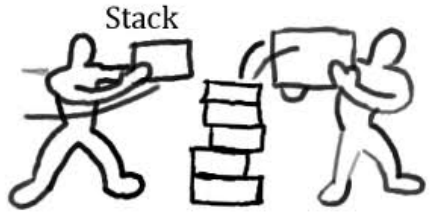
APUE 15.5



Introdução

- Os tubos só podem ser usados por processos que tenham um antecessor comum.
- Para resolver a limitação dos tubos, o Unix disponibiliza Filas (FIFO-"First In First Out"), também designados por tubos nomeados ("named pipes") que
  - podem ser usados por processos não relacionados entre si, e
  - são referenciandos por um identificador dentro do sistema de ficheiros.

Criação/Eliminação (1)



A. Criação de uma fila

1. Por chamada de sistema

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

– 1º parâmetro é o nome do ficheiro.

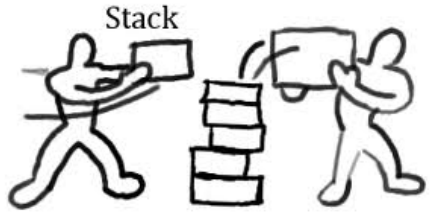
– 2º parâmetro identifica as permissões de acesso, iguais a qualquer ficheiro, determinados por OU de grupos de bits.

– As permissões de acesso também podem ser indicados por 3 dígitos octais, cada dígito 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/Eliminação (2)



2. Por comando Shell

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

- A fila é mantida no sistema de ficheiros até ser eliminada.

B. Eliminação de uma fila

1. Por chamada de sistema

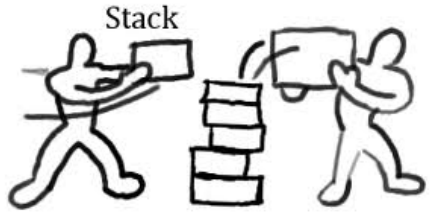
POSIX:
```
#include <unistd.h>
int unlink(char *);
```

– 1º parâmetro é o nome do ficheiro.

2. Por comando Shell

```
rm fichID
```

Criação/Eliminação (3)

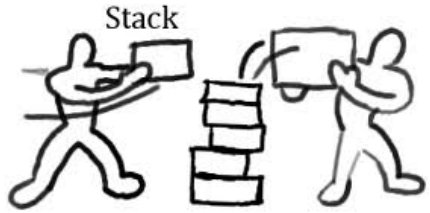


Exemplo:

Tipo de ficheiro é pipe

```
[rgc@asterix Comunicacao]$ mkfifo -m 644 tubo
[rgc@asterix Comunicacao]$ ls -l tubo
prw-r--r-- 1 rgc docentes 0 2010-09-26 13:49 tubo
[rgc@asterix Comunicacao]$ rm tubo
rm: remove fifo `tubo'? y
[rgc@asterix Comunicacao]$ ls -l tubo
ls: cannot access tubo: No such file or directory
[rgc@asterix Comunicacao]$
```

Abertura (1)

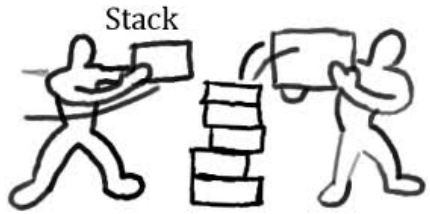


- Antes de ser usada, a fila tem de ser aberta pela chamada de sistema

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

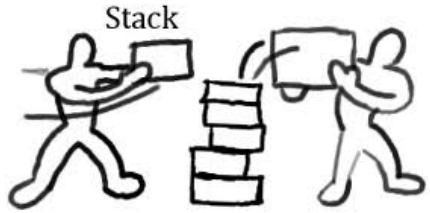
  – 1º parâmetro é o nome do ficheiro.
  – 2º parâmetro formado por bits que indicam:
    - Modos de acesso: `O_RDONLY` (leitura apenas) ou `O_RDONLY` (escrita apenas)
    - Opções de abertura: `O_CREAT` (criado se não existir), `O_NONBLOCK` (operação de E/S não são bloqueadas)
  – O valor de retorno é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

Abertura (2)



- Regras aplicadas na abertura de ficheios:
  - 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 com a excepção: se tiver sido indicada a opção `O_NONBLOCK`, o processo não fica bloqueado, sendo devolvido o valor -1 (`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 com a excepção: se tiver sido indicada a opção `O_NONBLOCK`, o processo não fica bloqueado, sendo devolvido o valor -1 (`errno` fica com valor `ENXIO`).

Leitura/escrita

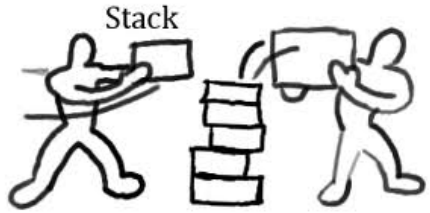


- Comunicação feita pelas mesmas chamadas de sistema dos tubos:

  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` (acção 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 (1)



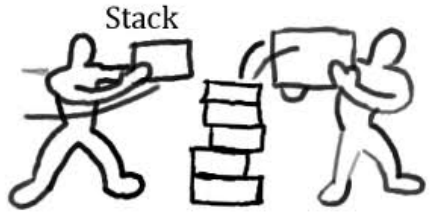
[Exemplo] Dois processos *writer* enviam mensagens para o processo *reader* através de uma fila.

- O identificador da fila e o comprimento da memória tampão é definida no ficheiro à parte.

defs.h

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

Exemplo : writer.c (2)

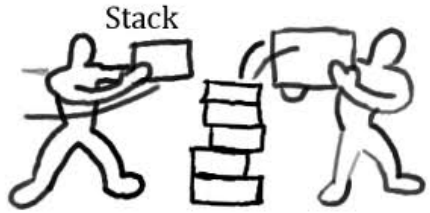



```
#include <stdio.h>
#include <string.h>
#include <sys/file.h>
#include "defs.h"
main() {
   int fd, i;
   char msg[LEN];

   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 : reader.c (3)

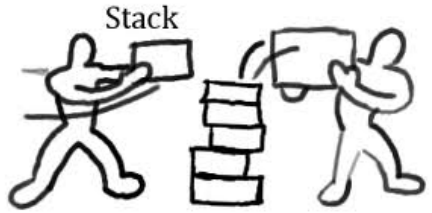



```
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/file.h>
#include "defs.h"

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 (4)



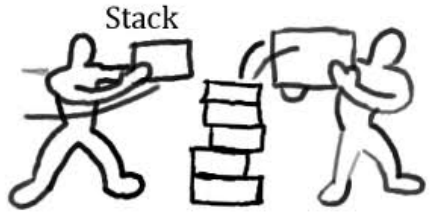

```
[rgc@asterix FIFO]$ reader & writer & writer &
[1] 7528
[2] 7529
[3] 7530
[rgc@asterix FIFO]$ Hello no 1 from process 7530
Hello no 1 from process 7529
Hello no 2 from process 7530
Hello no 2 from process 7529
Hello no 3 from process 7530
Hello no 3 from process 7529

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

Lança 1 leitor e 2 escritores

PIDs dos processos lançados

Exemplo (5)



- A lista não foi eliminada pelos programas

```
[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]$
```