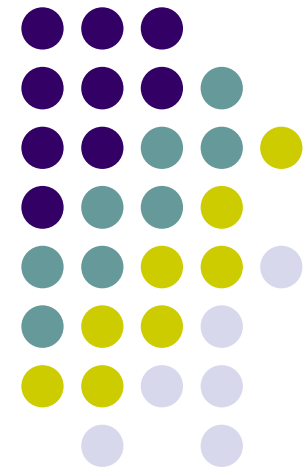
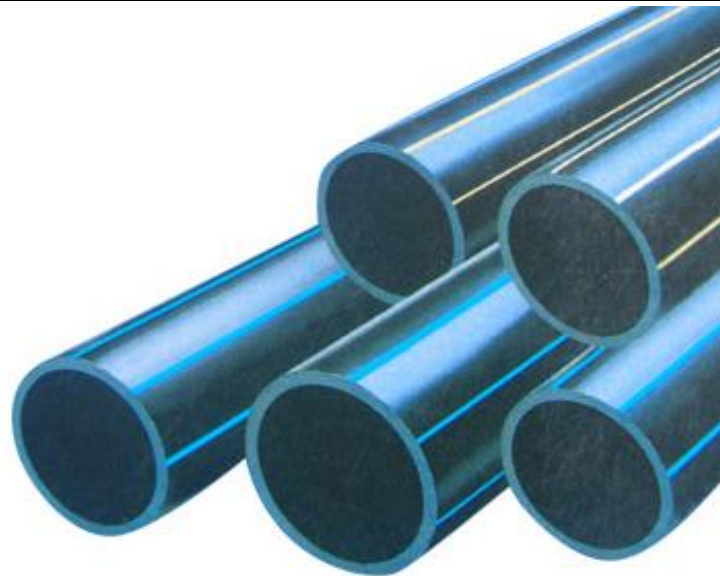


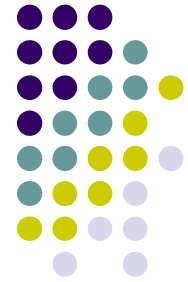
Comunicação entre Processos

Pipes

Redirecionamento de entrada e saída



Descritores de arquivos



- Cada processo tem sua própria tabela de descritores de arquivos para controlar todos os arquivos abertos

tabela do processo

	flags	ptr
fd 0:		
fd 1:		
fd 2:		
fd 3:		
	...	

tabela do arquivo

file status flag
current file offset
v-node ptr

tabela de v-node

Informação
tamanho do arquivo

tabela do processo

	flags	ptr
fd 0:		
fd 1:		
fd 2:		
fd 3:		
fd 4:		
	...	

cada processo tem seu
próprio ponteiro de
leitura/escrita de arquivo

file status flag
current file offset
v-node ptr

Descritores de arquivos



- Os descritores de arquivo 0, 1 e 2 se referem ao stdin (default: teclado), ao stdout (default: monitor) e stderr (default: monitor), respectivamente

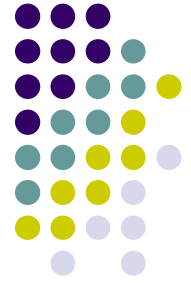
tabela do processo

	flags	ptr
fd 0:		
fd 1:		
fd 2:		
fd 3:		
	...	

stdin
stdout
stderr

- Os descritores de arquivos podem ser mudados para realizarem operações de comunicação inter-processos (operação de pipe)

Função pipe()



- Cria um canal de comunicação entre processos
- Definido em <unistd.h>

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

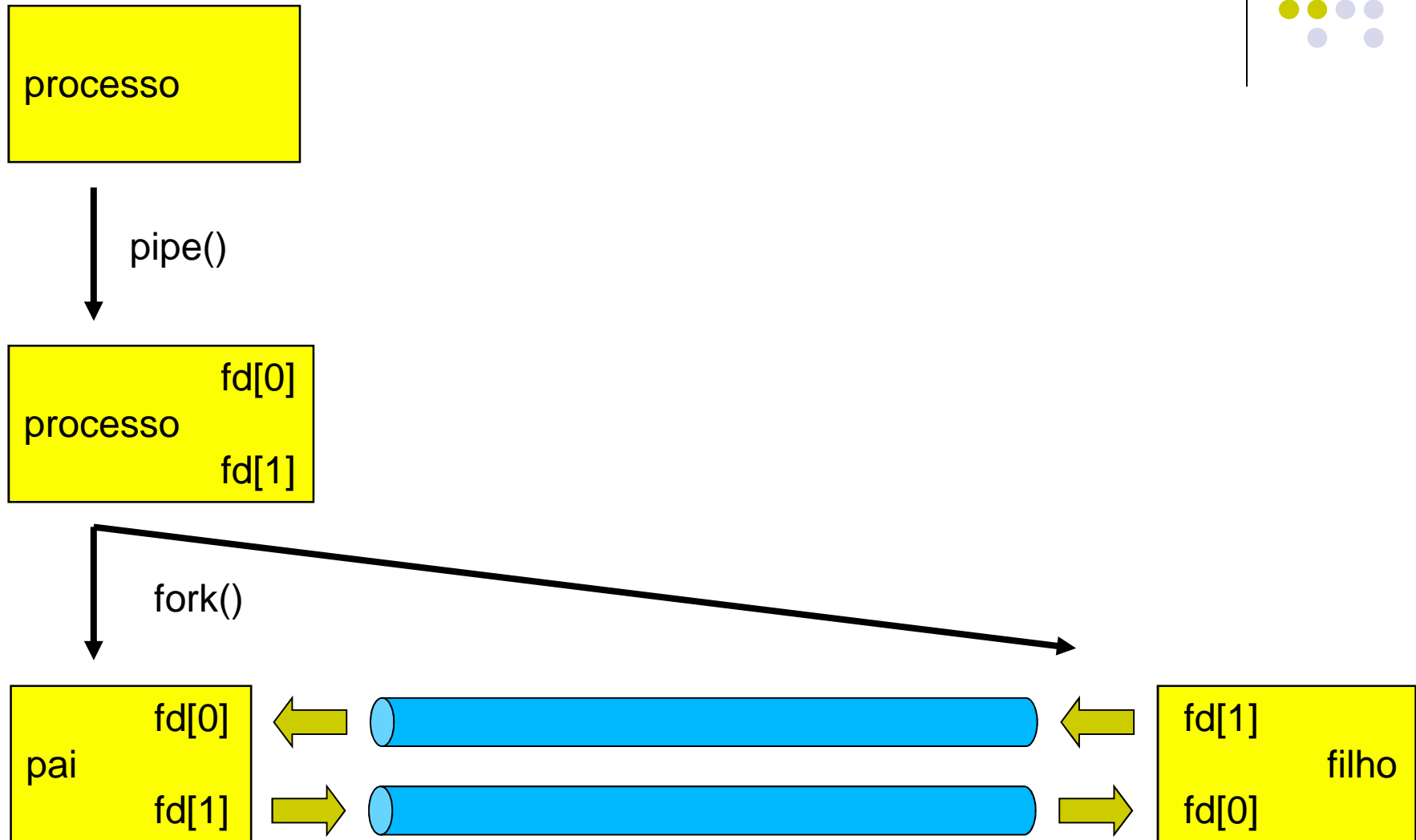
- Cria dois canais de comunicação:
 - fd[0] é aberto para leitura
 - fd[1] é aberto para escrita
- Retorna:
 - 0 (zero) em caso de sucesso
 - -1 em caso de erro

Função pipe()

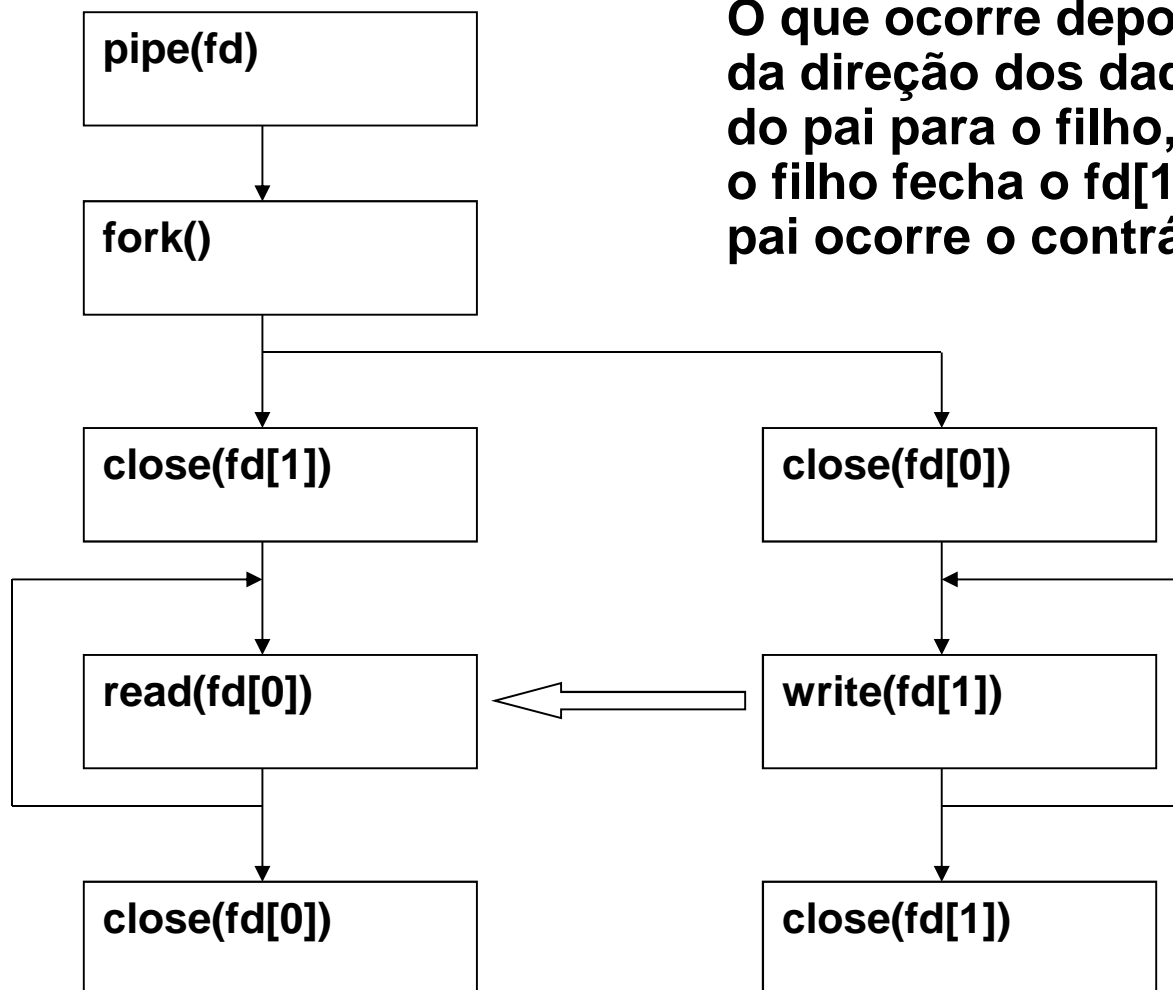


- Normalmente, após os pipes serem criados, dois ou mais processos colaborativos são criados através de ***fork()***
- Os dados são **transmitidos** e recebidos através de ***write()*** e ***read()***
- Pipes abertos pela função ***pipe()*** devem ser fechados pela função ***close()***
- Dados **escritos** no descritor de arquivo ***fd[1]*** podem ser **lidos** do ***fd[0]***
- Dois processos podem se comunicar através de um pipe se eles lêem e escrevem em ***fd[0]*** e ***fd[1]*** respectivamente

Função pipe()

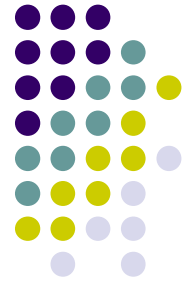


Esquema de comunicação via pipe



O que ocorre depois do fork depende da direção dos dados. Para um pipe do pai para o filho, o pai fecha `fd[0]` e o filho fecha o `fd[1]`. E do filho para o pai ocorre o contrário.

Criando os canais com pipe()



```
int fd[2]; // descritor dos pipes
```

```
if (pipe(fd) < 0)
{
    puts ("Erro ao abrir os pipes");
    exit (-1);
}
```

- Em caso de sucesso, a chamada à pipe() retorna 0 e fd[0] conterá o descritor de leitura e fd[1] o de escrita
- Em caso de falha, a função retorna -1

Função write()



- Utilizada para escrever dados em um arquivo ou qualquer outro objeto identificado por um descritor de arquivo (file descriptor)
- Definido em <unistd.h>

```
ssize_t write(int fildes, const void *buf, size_t nbyte);
```

- Onde
 - fildes : é o descritor do arquivo (ou do pipe)
 - buf : endereço da área de memória onde estão os dados que serão escritos
 - nbytes : número de bytes que serão escritos
- Valor retornado:
 - Em caso de sucesso, a função retorna a quantidade de dados escritos
 - Em caso de falha, o valor retornado difere da quantidade de bytes enviados

Função read()

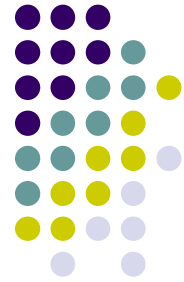


- Lê dados de um arquivo ou de qualquer outro objeto identificado por um descritor de arquivo
- Definido em <unistd.h>

```
ssize_t read(int fildes, void *buf, size_t nbyte);
```

- Onde:
 - fildes : descritor do arquivo
 - buf : endereço de memória onde os dados serão armazenados depois de lidos
 - nbyte : quantidade máxima de bytes que podem ser transferidos
- Retorna:
 - Quantidade de dados lidos

Exemplo do uso de pipe



```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>

int main (int argc, char *argv[])
{
    int nDadosTx, nDadosRx; // quantidade de dados transmitidos/recebidos
    int fd[2];              // descritor dos pipes
    const char textoTX[] = "uma mensagem";
    char textoRX[sizeof textoTX];

    if (pipe(fd) < 0)
    {
        puts ("Erro ao abrir os pipes");
        exit (-1);
    }

    nDadosTx = write(fd[1], textoTX, strlen(textoTX)+1);
    printf("%d dados escritos\n", nDadosTx);

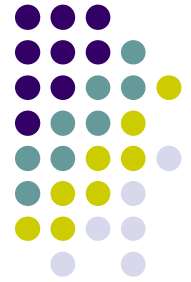
    nDadosRx = read(fd[0], textoRX, sizeof textoRX);
    printf("%d dados lidos: %s\n", nDadosRx, textoRX);

    close(fd[0]); close(fd[1]);

    return 0;
}
```

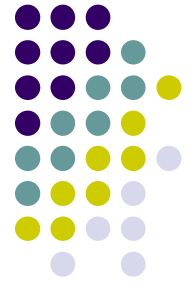
```
pipe$ make teste
gcc -g -Wall -o teste teste.c
pipe$ ./teste
13 dados escritos
13 dados lidos: uma mensagem
pipe$
```

Exemplo: pai escreve para o filho



```
void main ()
{
    int fd[2];
    pipe(fd);
    if (fork() == 0)
    { /* filho */
        close(fd[1]); /* fd[1] desnecessario */
        read(fd[0], ...); /* lê do pai */
        ...
    }
    else
    {
        close(fd[0]); /* fd[1] desnecessario */
        write(fd[1], ...); /* escreve para o filho */
        ...
    }
}
```

Funções dup() e dup2()



- Definidos em <unistd.h>
 - `int dup(int fd);`
 - Duplica o descritor de arquivo (fd) e armazena-o no descritor de menor número não usado do corrente processo
 - Pode ser usado para mudar o stdin (0) e o stdout (1)
 - `int dup2(int fd1, int fd2);`
 - Similar ao dup() mas com destino especificado
 - fd2 = valor do novo descritor
 - dup2 fecha fd2 antes de duplicar fd1
 - Retorna:
 - Em caso de sucesso: o número do descritor
 - Em caso de erro: -1

Funções dup() e dup2()



- Exemplo
 - novoFd = dup(1)

tabela do processo

	flags	ptr
fd 0:		
fd 1:		
fd 2:		
fd 3:		
	...	

tabela do arquivo

file status flag
current file offset
v-node ptr

tabela de v-node

Informação
tamanho do arquivo

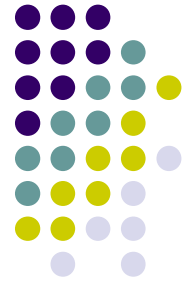
Funções dup() e dup2()



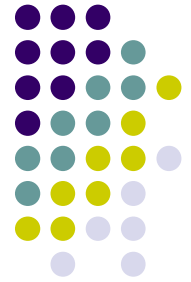
```
#include <...>
int main(void)
{
    int fd;          /* descritor a ser duplicado */
    int retorno;     /* valor de retorno de dup */
    int retorno2;     /* valor de retorno de dup2 */
    if ((fd=open("esteArquivo",O_RDWR|O_CREAT|O_TRUNC,0666)) == -1)
    {
        perror("Error open()");
        return -1;
    }
    close(0);        /* fechamento da entrada stdin */
    if ((retorno = dup(fd)) == -1)
    {
        /* duplicacao de stdin (menor descritor fechado) */
        perror("Error dup()");
        return -2;
    }
    if ((retorno2 = dup2(fd,1)) == -1)
    {
        /* duplicacao de stdout */
        perror("Error dup2()");
        return -3;
    }
    printf ("valor de retorno de dup(): %d \n",retorno);
    printf ("valor de retorno de dup2(): %d \n",retorno2);
    return 0;
}
```

```
pipe$ ./testeDup
pipe$ cat esteArquivo
valor de retorno de dup(): 0
valor de retorno de dup2(): 1
pipe$ █
```

Perguntas?



Exercícios



- 1) Faça um programa para criar dois processos que se comunicam via pipe. O Pai lê do pipe enquanto o filho escreve no pipe. Exiba o resultado do que foi escrito e lido.
- 2) Faça um programa para redirecionar a entrada e a saída, lendo os dados de um arquivo e gerando a saída em outro.
- 3) Faça um programa para criar um pipe e executar dois processos que são utilitários do Unix que se comuniquem através do pipe criado (a shell faz isso).