

```
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
```

```
#define MAXPAROLA 30
#define MAXRIGA 80
```

```
int main(int argc, char *argv[])
```

```
{
    int freq[MAXPAROLA]; /* vettore di contatori
delle frequenze delle lunghezze delle parole */
    char riga[MAXRIGA];
    int i, inizio, lunghezza;
    FILE * f;
```

```
    for(i=0; i<MAXPAROLA; i++)
        freq[i]=0;
```

```
    if(argc != 2)
```

```
    {
        fprintf(stderr, "ERRORE: serve un parametro con il nome del file\n");
        exit(1);
    }
```

```
    f = fopen(argv[1], "r");
    if(f==NULL)
```

```
    {
        fprintf(stderr, "ERRORE: impossibile aprire il file %s\n", argv[1]);
        exit(1);
    }
```

```
    while( fgets( riga, MAXRIGA, f ) != NULL )
```

Processes

Pipes and redirections Linux

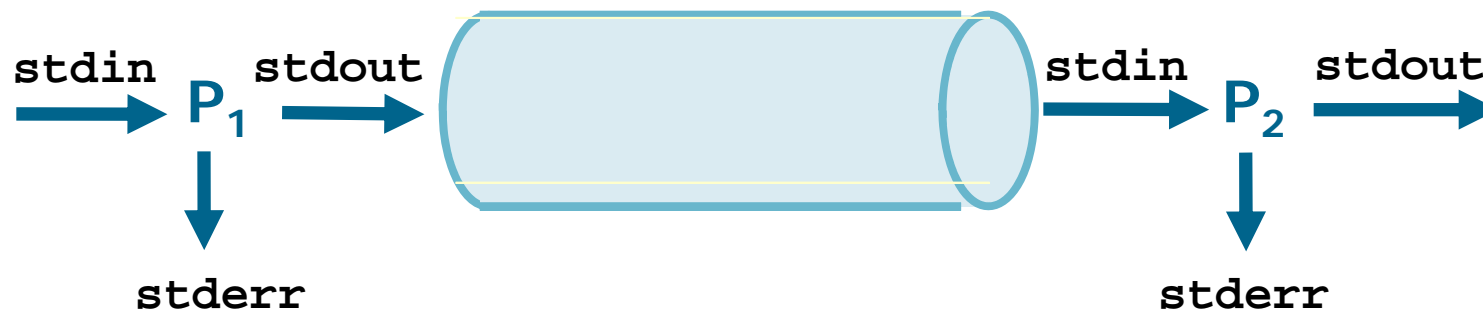
Stefano Quer and Pietro Laface

Dipartimento di Automatica e Informatica

Politecnico di Torino

Pipes

- ❖ Inter-process communication can be performed by processes executed by shell commands
- ❖ A shell **pipe** connects the standard output of a sender process, and the standard input of a receiving process



Pipe

`command1 | command2`

❖ Examples

- `ls -la | more`
- `ps | grep main`
- `cat file1.txt file2.txt file3.txt | sort`
- `ls -laR *.c | wc`

I/O redirection

- ❖ The standard files

- Standard input (stdin, 0)
- Standard output (stdout, 1)
- Standard error (stderr, 2)

can be redirected by the shell

- ❖ A process reads/writes data from a source/destination different with respect to the predefined standard ones

I/O redirection

- ❖ A special file
 - `/dev/null`
- ❖ Writing on `/dev/null` does not produce any output (`/dev/null` is a sink)
- ❖ Reading from `/dev/null` returns a sequence of zeros

Standard input

```
command < file
```

- ❖ Standard input redirection (reads from a file)

```
command << marker  
... text ...  
marker
```

- ❖ Standard input redirection (reads from terminal)
 - "here document"
 - marker is a generic string
 - Often **EOF**

Standard output

```
command > file  
command 1> file
```

❖ Standard output redirection on a file

- If the file exist it is overwritten
- Descriptor 1 (stdout) is the default
 - Thus it is normally omitted

```
command >> file
```

❖ Standard output redirection on a file (append)

Standard error

```
command 2> file
```

```
command 2>> file
```

- ❖ Standard error redirection on a file
- ❖ Standard error redirection on a file (append)

Both streams

`command &> file`

`command &>> file`

& is not the last character of the line !!

- ❖ Standard **and** error redirection on a file
- ❖ Standard **and** error redirection on a file (append)

Multiple redirection

```
command 1> fileOut 2> fileErr  
command > fileOut 2> fileErr
```

- ❖ Redirection on different files of
 - ❖ Standard output
 - ❖ Standard error