

```
#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

## Shell commands for Pipes and redirections

Stefano Quer, Pietro Laface, and Stefano Scanzio

Dipartimento di Automatica e Informatica

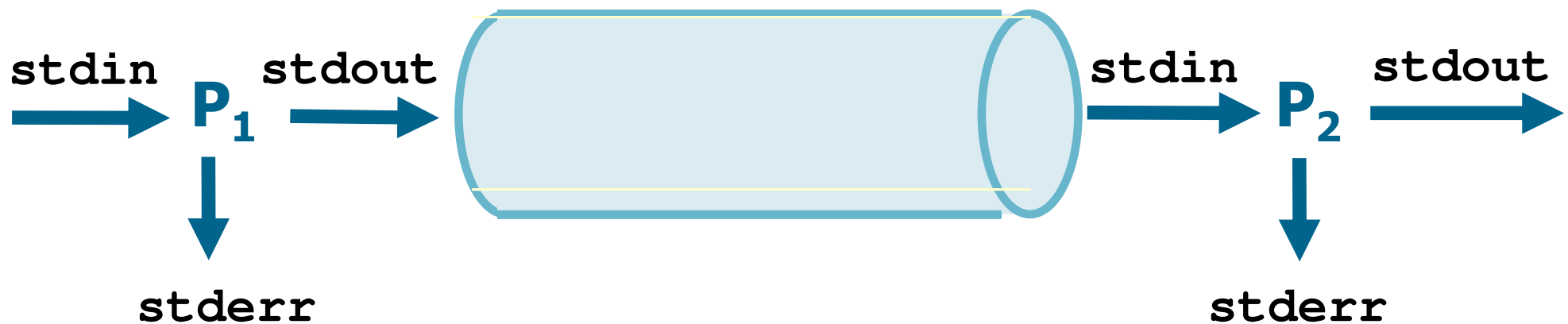
Politecnico di Torino

[skenz.it/os](http://skenz.it/os)

[stefano.scanzio@polito.it](mailto:stefano.scanzio@polito.it)

# Pipes

- ❖ Inter-process communication can be performed also 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  
command1 | command2 | command3 ...
```



## ❖ Examples

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

## I/O redirection

- ❖ The term redirection indicates the deviation of the standard channels, i.e.
  - Standard input (stdin, 0)
  - Standard output (stdout, 1)
  - Standard error (stderr, 2)
- ❖ In practice, a process (a command) 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)

```
ls -laR > file_out.  
wc prgm.c > file_pout.txt
```

```
ls -laR >> file_out.  
wc prgm.c >> file_pout.txt
```

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

## Multiple redirection

### Bash Shell

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

### Tcsh Shell

```
command > fileOut >& fileErr
```

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

# Example

Redirection of  
stdin, stdout,  
stderr

```
int main () {  
    char c;  
    setbuf(stdout, 0);  
    setbuf(stderr, 0);  
    while (scanf ("%c", &c) == 1) {  
        fprintf (stdout, "stdout:%c\n", c);  
        fprintf (stderr, "stderr:%c\n", c);  
    }  
    return (0);  
}
```

## ❖ Redirection

```
comando | pgrm  
pgrm < file  
pgrm > file  
pgrm 1> fileOut 2> fileErr  
pgrm < fileIn 1> fileOut 2> fileErr
```