

```
#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 )
```

Processi

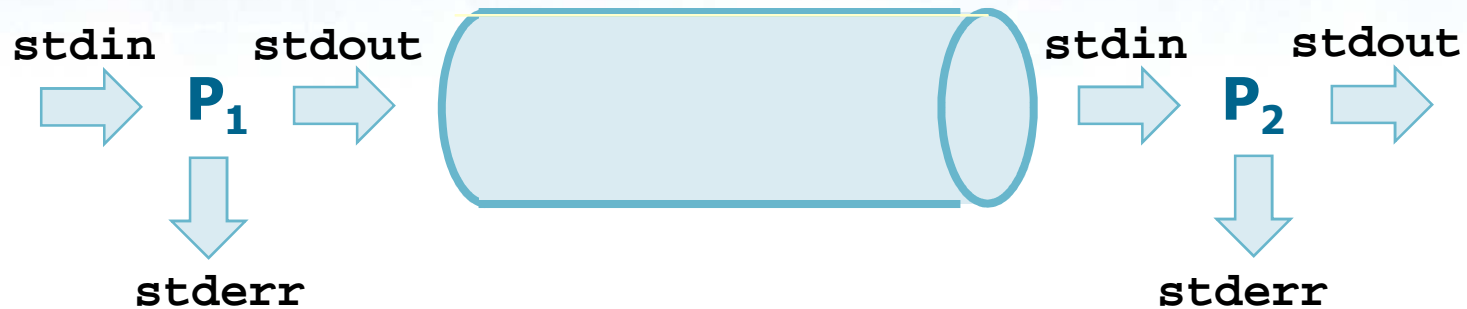
Pipe e ridirezione in UNIX/Linux

Stefano Quer

Dipartimento di Automatica e Informatica

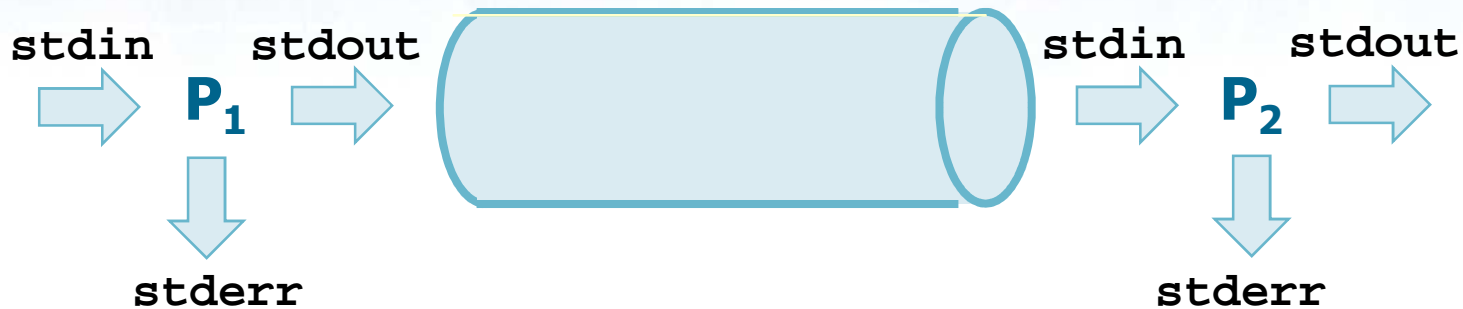
Politecnico di Torino

Pipe



- ❖ La comunicazione tra processi attraverso pipe può essere realizzata anche quando i processi sono eseguiti mediante comandi di shell
- ❖ Le pipe a livello di comandi di shell si rappresentano con il carattere “|”

Pipe

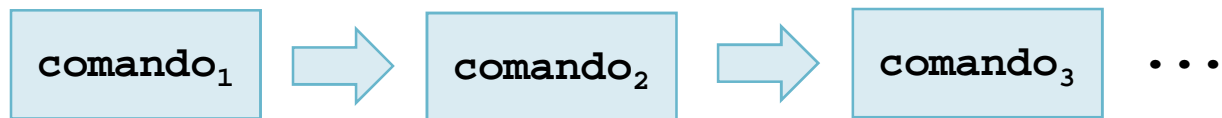


❖ Una pipe

- Crea un collegamento tra lo standard output del comando precedente e lo standard input di quello successivo
- Un pipe risiede completamente in memoria
 - La comunicazione è più veloce che quella che avviene attraverso disco e R/W da/su file

Pipe

```
comando1 | comando2  
comando1 | comando2 | comando3 ...
```



❖ Esempi

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

`ls -laR *.c | wc`
versus
`wc *.c`

Redirezione dell'I/O

- ❖ Il termine redirezione indica la deviazione dei dispositivi standard su canali diversi
 - Standard input (stdin, 0)
 - Standard output (stdout, 1)
 - Standard error (stderr, 2)
- ❖ In pratica un dato comando legge/scrive dati da sorgente/destinazione diversa da quelle predefinite

Redirezione dell'I/O

- ❖ Si osservi che il dispositivo
 - /dev/null
 - È un file speciale su cui si può
 - Scrivere senza accumulare byte su disco o visualizzarli a video
 - Leggere una sequenza di valori 0

Standard input

```
comando < file
```

- ❖ La redirectione dello standard input da un file si effettua con il carattere ``<``

Standard input

```
comando << marker  
... testo ...  
marker
```

EOF

EOF

- ❖ Redirezione dello standard input da tastiera
 - "Documento sul posto" ("here document")
 - Marker è una stringa arbitraria
 - Normalmente EOF

Standard output

```
comando > file  
comando 1> file
```

Default: 1> → >

❖ Redirezione dello standard output su un file

- Se il file esiste viene cancellato
- Il descrittore 1 (stdout) è quello di default
 - Quindi normalmente viene omesso
- Esempi
 - `ls -lar > fileOut`
 - `wc pgrm.c > fileOut`

Standard output

```
comando >> file
```

- ❖ Accodamento dello standard output su un file
 - Se il file esiste **non** viene cancellato
 - Esempi
 - `ls -lar >> fileOut`
 - `wc pgrm.c >> fileOut`

Standard error

```
comando 2> file
```

- ❖ Redirezione dello standard error sul file
 - Analogo allo stdout ma su stderr

Standard error

```
comando 2>> file
```

- ❖ Accodamento dello standard error sul file
 - Analogo allo stdout ma su stderr

Redirezione multipla

- ❖ Redirezione contemporanea
 - Dello standard output
 - Dello standard error
- ❖ Sullo stesso file

```
comando &> file  
comando &>> file
```

- ❖ Su file diversi

```
comando 1> fileOut 2> fileErr  
comando > fileOut 2> fileErr
```

Esempio

Redirezione di
stdin, stdout,
stderr

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

❖ Redirezione

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