



# Sistemi Operativi<sup>1</sup>

Mattia Monga

Dip. di Informatica e Comunicazione  
Università degli Studi di Milano, Italia  
[mattia.monga@unimi.it](mailto:mattia.monga@unimi.it)

a.a. 2011/12

<sup>1</sup> © 2012 M. Monga. Creative Commons Attribuzione-Condividi allo stesso modo 2.5 Italia License.  
<http://creativecommons.org/licenses/by-sa/2.5/it/>. Immagini tratte da [?] e da Wikipedia.



## Lezione VIII: Shell 2



## shell (pseudo codice)

```

1 while (1){ /* repeat forever */
2     type_prompt(); /* display prompt on the screen */
3     read_command(command, parameters); /* read input from terminal */
4     if (fork() > 0){ /* fork off child process */
5         /* Parent code. */
6         waitpid(1, &status, 0); /* wait for child to exit */
7     } else {
8         /* Child code. */
9         execve(command, parameters, 0); /* execute command */
10    }
11 }

```



## Esercizi

- 1 Scrivere, compilare (`cc -o nome nome.c`) ed eseguire un programma che *forca* un nuovo processo.
- 2 Scrivere un programma che stampi sullo schermo ‘Hello world! (numero)’ per 10 volte alla distanza di 1 secondo l’una dall’altra (`sleep(int)`). Terminare il programma con una chiamata `exit(0)`
- 3 Usare il programma precedente per sperimentare l’esecuzione in sequenza e in parallelo
- 4 Controllare il valore di ritorno con `/bin/echo $?`
- 5 Tradurre il programma in assembly con `cc -S nome.c`
- 6 Modificare l’assembly affinché il programmi esca con valore di ritorno 3 e controllare con `echo $?` dopo aver compilato con `cc -o nome nome.s`
- 7 Modificare l’assembly in modo che usi `scanf` per ottenere il numero di saluti.

## Pipe



Sistemi Operativi

Bruschi Monga

Shell  
Esercizi  
Shell programming

172

```
ls | sort
1 int main(void){
2     int fd[2], nbytes; pid_t childpid;
3     char string[] = "Hello, world!\n";
4     char readbuffer[80];
5
6     pipe(fd);
7     if(fork() == 0){
8         /* Child process closes up input side of pipe */
9         close(fd[0]);
10        write(fd[1], string, (strlen(string)+1));
11        exit(0);
12    } else {
13        /* Parent process closes up output side of pipe */
14        close(fd[1]);
15        nbytes = read(fd[0], readbuffer, sizeof(readbuffer));
16        printf("Received string: %s", readbuffer);
17    }
18    return(0);}
```

## Pipe (cont.)



Sistemi Operativi

Bruschi Monga

Shell  
Esercizi  
Shell programming

173

```
1     if(fork() == 0)
2     {
3         /* Close up standard input of the child */
4         close(0);
5
6         /* Duplicate the input side of pipe to stdin */
7         dup(fd[0]);
8         execlp("sort", "sort", NULL);
9     }
```

## Un vero linguaggio di programmazione



Sistemi Operativi

Bruschi Monga

Shell  
Esercizi  
Shell programming

174

La shell è un vero (Turing-completo) linguaggio di programmazione (interpretato)

- Variabili (create al primo assegnamento, uso con \$, export in un'altra shell).
  - x="ciao"; y=2 ; /bin/echo "\$x \$y \$x"
- Istruzioni condizionali (valore di ritorno 0 ~> true)
  - if /bin/ls piripacchio; then /bin/echo ciao; else /bin/echo buonasera; fi
- Iterazioni su insiemi
  - for i in a b c d e; do /bin/echo \$i; done
- Cicli
  - /usr/bin/touch piripacchio
  - 2 while /bin/ls piripacchio; do
  - 3 /usr/bin/sleep 2
  - 4 /bin/echo ciao
  - 5 done & ( /usr/bin/sleep 10 ; /bin/rm piripacchio )