Lezione 10

- Scrivere un programma C che prende in input sulla linea di comando tre parametri `file1`, `file2`, `file3` ed esegue la seguente lista di comandi Unix:

```
cp file1 file2
sort file2 -o file3
cat file3
```

```
#include <stdio.h>
#include <unistd.h>
#include <sys/wait.h>

main(int argc,char **argv) {
  pid_t pid;

  if(argc!=4) {
    fprintf(stderr,"Uso del programma: prog <file1> <file2> <file3>\n");
    exit(1);
  }

  pid=fork();

  switch(pid) {
  case -1:
    perror("Errore nella chiamata a fork");
    exit(2);
  case 0:
    execlp("cp","cp",argv[1],argv[2],NULL);
    perror("Errore nella chiamata a execlp");
    exit(3);
  default:
    waitpid(pid,NULL,0);
    pid=fork();

    switch(pid) {
    case -1:
      perror("Errore nella chiamata a fork");
      exit(2);
    case 0:
      execlp("sort","sort",argv[2],"-o",argv[3],NULL);
      perror("Errore nella chiamata a execlp");
      exit(3);
    default:
      waitpid(pid,NULL,0);
      pid=fork();

      switch(pid) {
      case -1:
        perror("Errore nella chiamata a fork");
```

```
        exit(2);
      case 0:
        execlp("cat","cat",argv[3],NULL);
        perror("Errore nella chiamata a execlp");
        exit(3);
      default:
        waitpid(pid,NULL,0);
        exit(0);
      }

    }

  }

}
```

- Scrivere un programma C (chiamandolo `run.c`) che esegue quanto passato sulla linea di comando. Ad esempio:

```
> run ls -l
> run wc -c
...
```

```
#include <stdio.h>
#include <unistd.h>
#include <sys/wait.h>

main(int argc,char **argv) {
  pid_t pid;

  if(!(argc>1)) {
    fprintf(stderr,"Uso del programma: run <comando>\n");
    exit(1);
  }

  pid=fork();

  switch(pid) {
  case -1:
    perror("Errore nella chiamata a fork");
    exit(2);
  case 0:
    execvp(argv[1],&argv[1]);
    perror("Errore nella chiamata a execvp");
    exit(3);
  default:
    waitpid(pid,NULL,0);
    exit(0);
  }

}
```