Laboratorio di Sistemi Operativi: Prof.ssa A. Rescigno **Anno Acc. 2010-2011**

Prova in itinere 21 Aprile 2011

Università di Salerno

Nome e Cognome: Matricola:

1	2	3	4	5	6	tot
/18	/22	/12	/20	/12	/16	/100

Spazio riservato alla correzione

1. *18 punti*
   Sia `pippo` un file che contiene la frase `Esame di Sistemi Operativi` e sia `prog1.out` l'eseguibile corrispondente al codice sottostante:

```
#include<stdio.h>
#include <fcntl.h>
#include <unistd.h>
int main()
{
(1)   fd = open("pippo", O_APPEND|O_WRONLY);
(2)   close(1);
(3)   if (dup(fd) >= 0)
(4)      {
(5)            write(1,"Primo Anno",10);
(6)         }
(7)  }
```

   a) dire che cosa succede mandando in esecuzione `prog1.out` e spiegare dettagliatamente il perché;

b) spiegare quale potrebbe essere la situazione in cui nella riga (3) si ottiene -1 dalla chiamata di `dup(fd)`; dare la soluzione che risolverebbe il problema.

c) sia `prog2.out` l'eseguibile corrispondente al codice sottostante:

```
#include<stdio.h>
#include <fcntl.h>
#include <unistd.h>
int main()
{
(1)   fd = open("pippo", O_APPEND|O_WRONLY);
(2)   close(1);
(3)   if (dup(fd) >= 0)
(4)      {
(5)           printf(" Primo\n");
(6)           write(1,"Anno",4);
(7)         }
(8)  }
```

spiegare perché, mandando in esecuzione `prog2.out`, si ottiene un risultato diverso rispetto a quanto ottenuto mandando in esecuzione `prog1.out`.

2. *22 punti*
   (a) *(12punti)* Sia `File` un file lungo 20 byte.

   ```
   -rw------- 1 rescigno 20  Jun 4 09:45 File
   ```

   Scrivere un programma C in cui
   (1) - si crei un hard link `HFile` ed un symbolic link `SFile` a `File`,
   (2) - si visualizzi sullo standard-output il contenuto di `File` in modo tale che i primi 10 byte di `File` siano visualizzati **utilizzando** `HFile` e i secondi 10 byte di `File` siano visualizzati **utilizzando** `SFile`;
   (3) - si crei un hard link `HHFile` ad `HFile`;
   (4) - si visualizzi sullo standard output il numero di link di `File`, `HFile`, `HHFile` e `SFile`.

(b) *(5 punti)* Supponendo di aggiungere al codice precedente anche le istruzioni seguenti:

```
chmod(HFILE, 0600);
chmod(HHFILE, 0400);
chmod(SFILE, 0200);
```

dire quali saranno i permessi dei file `File`, `HFile`, `HHFile` e `SFile` dopo aver mandato in esecuzione l'eseguibile con i cambiamenti sopra riportati e spiegare il perché.

(b) *(5 punti)* Disegnare la Process Table, la File Table e la v-node Table relative all'esecuzione del programma del punto (a).

3. *12 punti*
   Si consideri il seguente programma e si supponga di compilarlo.

```
#include<sys/types.h>
#include<fcntl.h>
#include<unistd.h>

int main(){
        if (access("tentativo.txt", O_WRONLY)<0)
                        printf("access error per prova.txt");
         else
            printf("access OK\n");

    if (open("tentativo.txt", O_WRONLY)<0)
                       printf("open error per prova.txt");
         else
     printf("open OK\n");

        if (open("prova.txt", O_WRONLY)<0)
                       printf("open error per prova.txt");
         else
     printf("open OK\n");

        exit(0);
}
```

   Se fosse

```
-r-xr-xr-x 1 rescigno 10932 Jun 4 10:45 a.out
-rw------- 1 rescigno 1891  Jun 4 09:45 prova.txt
-rw------- 1 straniero 1891  Jun 4 09:45 tentativo.txt
```

   supponendo che si sia loggato `straniero`, dire

   1) che cosa succede dando `a.out`.

   2) Si assuma ora di settare il set-user-id di `a.out`. Dire se ci sono cambiamenti dando `a.out`.

   3) supponendo ora che si sia loggato `studente` ripetere i passi 1) e 2).

   In tutti i casi le risposte vanno giustificate.

4. *20 punti*
   (a) Scrivere un programma C in cui, dato l'intero N da linea di comando, 2^N processi scrivano il proprio pid sullo standard output.

(a) Scrivere un programma C in cui, dato l'intero N da linea di comando, N processi scrivano il proprio pid sullo standard output.

5. *12 punti*
   Dato il seguente programma C, il cui eseguibile é `a.out`

```
(1)   void handler(int);
(2)   void exit1(void);
(3)   void exit2(void);

(4)   int main(void)
(5)   { char arry[6]="Hello ";
(6)     atexit(exit1);
(7)     if (fork()==0)
(8)                 atexit(exit2);
(9)   else  wait( );

(10)    printf("Ciao\n");
(11)     write(1,arry,6);
(12)    exit(0);}

(13)  void exit1(void)
(14)  { printf("Exit Handler 1\n"); }
(15)  void exit2(void)
(16)  { printf("Exit Handler 2\n"); }
```

   (a) dire che cosa succede dando `a.out`. Motivare la risposta.

   (b) dire che cosa contiene `FILE` dando `a.out > FILE`. Motivare la risposta.

   (c) modificando la linea (12) con `_exit(0);` dire che cosa succede dando `a.out`. Motivare la risposta.

6. *16 punti*
   Si scrivano **due** programmi C che, **utilizzando** una delle funzioni `exec`, simulino il comando bash `ls > FILE`, dove `FILE` non esiste.

FOGLIO DA UTILIZZARE PER LA BRUTTA

FOGLIO DA UTILIZZARE PER LA BRUTTA

FOGLIO DA UTILIZZARE PER LA BRUTTA

FOGLIO DA UTILIZZARE PER LA BRUTTA

FOGLIO DA UTILIZZARE PER LA BRUTTA