Università di Salerno - Dipartimento di Informatica
Tutorato di Sistemi Operativi a.a. 2017/2018
Federico Vastarini

Esercizi sull'utilizzo della funzione fork()

1. Eseguendo questo codice quante volte verrà mostrata la scritta "forked"?

```
. . .
printf("forked\n") ;
fork() ;
printf("forked\n") ;
. . .
```

2. Ed in questo caso?

```
. . .
fork() ;
printf("forked\n") ;
printf("forked\n") ;
fork() ;
printf("forked\n") ;
. . .
```

3. Ed in questo?

```
. . .
printf("forked\n") ;
fork() ;
fork() ;
printf("forked\n") ;
printf("forked\n") ;
. . .
```

4. E questo?

```
. . .
```

```
for (i = 0 ; i < 3 ; ++i)
     {
     fork() ;
     }
printf("forked\n") ;
. . .
```

5. E... questo?

```
. . .
printf("forked\n") ;
for (i = 0 ; i < 3 ; ++i)
     {
     fork() ;
     printf("forked\n") ;
     }
. . .
```

6. E... quest'ultimo?

```
. . .
for (i = 2 ; i < 4 ; ++i)
     {
     printf("forked\n") ;
     fork() ;
     printf("forked\n") ;
     }
. . .
```

7. In che ordine verranno mostrate le lettere a,b e c eseguendo il seguente codice?

```
. . .
write(0, "a", 1) ;
fork() ;
write(0, "b", 1) ;
fork() ;
write(0, "c", 1) ;
. . .
```

8. Nel seguente codice la parola "forked" viene mostrata più volte dal processo parent o da quello child?

```
. . .
printf("forked") ;
pid = fork() ;
if (!pid)
     printf("forked") ;
printf("forked") ;
. . .
```

9. Qual è il totale dei child process generati dalla funzione fork() nel seguente codice?

```
. . .
for (i = 0 ; i < 100 ; ++i)
     {
     pid = fork() ;
     if (!pid)
          printf("I am a forked process!") ;
     else
          break ;
     }
. . .
```

10. Qual è la differenza fondamentale tra l'esecuzione del seguente codice ed il precedente?

```
. . .
for (i = 0 ; i < 100 ; ++i)
     {
     pid = fork() ;
     if (!pid)
          {
          printf("I am a forked process!") ;
          break ;
          }
     }
. . .
```

11. Quale sarà l'output complessivo dell'esecuzione del seguente codice?

```
. . .
int x = 0 ;
pid = fork() ;
if (!pid)
     x++ ;
else
     x-- ;
printf("x=%d\n", x) ;
. . .
```

Soluzioni:

1. 3 volte. La prima dal processo principale ed un'altra da ognuno dei due processi generati dalla fork().
2. 8 volte. Due volte per ognuno dei due processi generati dalla prima fork() ed un'altra per ognuno dei quattro processi generati dalla seconda fork().
3. 9 volte. La prima dal processo principale ed altre due per ognuno degli otto processi generati dalla coppia di fork().
4. 8 volte (pari al numero totale di processi e del numero delle foglie dell'albero generato).
5. 15 volte (pari al numero di nodi dell'albero generato, ovvero $2^n - 1$).
6. 9 volte. Ogni volta che entra nel ciclo for mostra "forked" 3 volte. Entra nel ciclo 3 volte poiché dopo la prima fork() anche l'ultimo passo del ciclo verrà duplicato.
7. Verrà prima mostrata la lettera "a", ma non c'è modo di stabilire in che ordine verranno mostrate le due "b" e le quattro "c".
8. 2 volte dal parent e 2 volte dal child.
9. 100.
10. In questa versione del codice i processi non sono eseguiti in ordine mentre nella precedente sì.
11. Ci saranno due output differenti: x=-1 ed x=1. Si ricorda che i processi parent e child condividono lo stesso indirizzo fisico della variabile fin quando non ci sarà un accesso alla stessa in scrittura.