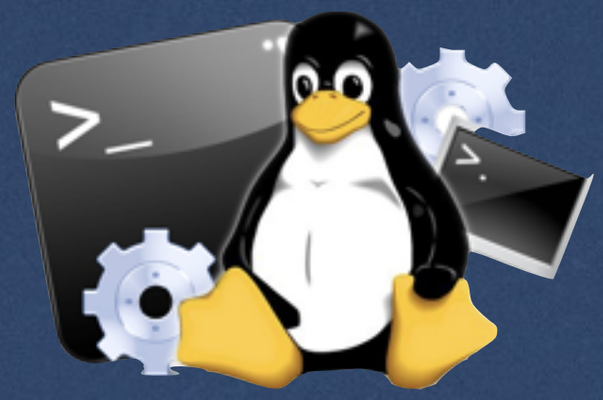


Final Exercises

Roberto De Virgilio

Sistemi operativi - 16 Gennaio 2017

Esercizio (espressioni regolari)

- *Dato il file di testo esempio.txt*

```
15 fifteen

14 fourteen

13 thirteen

12 twelve

.............

1  one
```

- *scrivere un comando linux per mostrare tutte le righe di esempio.txt*

Esercizio (espressioni regolari)

- *grep '.*' esempio.txt*

```
15 fifteen

14 fourteen

13 thirteen

12 twelve

.....

1  one
```

Esercizio (espressioni regolari)

- *Dato il file di testo esempio.txt*

```
15 fifteen

14 fourteen

13 thirteen

12 twelve

...........

1  one
```

- *scrivere un comando linux per mostrare tutte le righe di esempio.txt contenenti i caratteri 2 oppure 4 indipendentemente dalla loro collocazione nella stringa*

Esercizio (espressioni regolari)

- *grep '[24]' esempio.txt*

```
14 fourteen

12 twelve

4 four

2 two
```

Esercizio (espressioni regolari)

- *Dato il file di testo esempio.txt*

```
15 fifteen

14 fourteen

13 thirteen

12 twelve

.............

1  one
```

- *scrivere un comando linux per mostrare tutte le righe di esempio.txt che contengono due "e" di seguito, seguite da un carattere qualsiasi.*

Esercizio (espressioni regolari)

- *grep 'ee.' esempio.txt*
- *grep -E 'e{2}.' esempio.txt*

```
14 fourteen

12 twelve

4 four

2 two
```

Esercizio (espressioni regolari)

- *Dato il file di testo esempio.txt*

```
15 fifteen

.............

1   one
```

- *scrivere un comando linux per mostrare tutte le le righe che:*
  - *Iniziano con il carattere "1"*
  - *Terminano con il carattere "e"*
  - *Presentano al loro interno zero o più caratteri*

Esercizio (espressioni regolari)

- `grep '^1.*e$' esempio.txt`

```
12 twelve

1 one
```

Esercizio (espressioni regolari)

- *Dato il file di testo esempio.txt*

```
15 fifteen

.............

1   one
```

- *scrivere un comando linux per mostrare tutte le le righe che:*
  - *Iniziano con una qualsiasi sequenza di caratteri*
  - *deve essere presente una parola che inizia con la lettera 'f' seguita da un carattere compreso tra 'i' e 'p'*

Esercizio (espressioni regolari)

- `grep -E '^.*\<f[i-p]' esempio.txt`

```
15 fifteen

14 fourteen

5 five

4 four
```

Esercizio (espressioni regolari)

- *Dato il file di testo esempio.txt*

```
15 fifteen

.....

1   one
```

- *scrivere un comando linux per rimuove i primi 3 caratteri ad inizio linea*

Esercizio (espressioni regolari)

- *sed 's/...//' esempio.txt*

```
fifteen

fourteen

thirteen

twelve

…

wo

ne
```

Esercizio (AWK)

- *Si consideri il file 1975.txt che, riga per riga, contiene: nome di una nazione, la sua superficie in migliaia di km2, la sua popolazione in milioni di abitanti e il continente di riferimento:*

```
USSR        86250    262      Asia

USA         3615     219      America

Cina        3692     866      Asia

Canada      3852     24       America

Brasile     3286     116      America

Australia   2968     14       Oceania

India       1269     637      Asia

Argentina   1072     26       America

Sudan       968      19       Africa

Algeria     920      18       Africa
```

Esercizio (AWK)

- *scrivere uno script basato sull'utilizzo di AWK che stampi esclusivamente nazione e continente.*
- *scrivere uno script basato sull'utilizzo di AWK che stampi solo i dati relativi alle nazioni asiatiche.*
- *scrivere uno script basato sull'utilizzo di AWK che stampi Continente - Numero Medio di Abitanti (calcolato come somma del numero di abitanti diviso il numero di nazioni presenti nel continente).*

Esercizio (AWK)

- *Creiamo il programma* **AWK** *che stampi esclusivamente nazione e continente:*

```
#!/bin/awk -f

 BEGIN {
   printf(“[Nazione] [Continente]\n”);
 }
 {
   printf("%s %s\n”,$1,$4);
 }
```

Esercizio (AWK)

- *Creiamo il programma* **AWK** *che stampi solo i dati relativi alle nazioni asiatiche:*

```
#!/bin/awk -f

 BEGIN {

   printf("[Nazioni Asiatiche]\n");

   printf("[nazione] [superficie] [popolazione]\n”);

 }

($4 == "Asia") {

   printf("%s %d %d\n",$1,$2,$3);

 }
```

Esercizio (AWK)

- *Creiamo il programma* **AWK** *che stampi Continente - Numero Medio di Abitanti:*

```
#!/bin/awk -f
 BEGIN {
        printf("[Continente] [Numero Medio Abitanti]\n");
      }
{
        abitanti[$4] += $3;
        nazioni[$4] += 1;
 }
END {
    for (i in abitanti) {
        printf("%s %lf\n",i, abitanti[i]/nazioni[i]);
        }
    }
```

Esercizio (Debugging)

- *Si esegua il debug del seguente file Concatena.c per capire quale problema comporta al suo interno*

```
#include <string.h>
#include <stdio.h>
#define BUFLEN 16


int main() {
    char s1[BUFLEN], s2[BUFLEN];
    strcpy(s1,  "This is source");
    sprintf(s2, "%s", "This is destination");
    strcat(s2, s1);
    printf("Final string: |%s|", s2);
    return 0;
}
```

Esercizio (Debugging)

- *Si esegua il debug del seguente file Concatena.c per capire quale problema comporta al suo interno*

```
gcc -Wall -g concat.c -o concat

./concat

Abort trap: 6

gdb concat

(gdb) run

Starting program: /Users/rdevirgilio/Desktop/concat

Program received signal SIGABRT, Aborted.

0x00007fff874dcf06 in ?? () from /usr/lib/system/
libsystem_kernel.dylib
```

Esercizio (Debugging)

- *Si esegua il debug del seguente file Concatena.c per capire quale problema comporta al suo interno*

```
(gdb) back
#0  0x00007fff874dcf06 in ?? () from /usr/lib/system/
libsystem_kernel.dylib
#1  0x00007fff839614ec in pthread_kill ()
   from /usr/lib/system/libsystem_pthread.dylib
#2  0x00007fff886ea6df in abort () from /usr/lib/system/
libsystem_c.dylib
#3  0x00007fff886ea856 in abort_report_np ()
   from /usr/lib/system/libsystem_c.dylib
#4  0x00007fff88710a0c in __chk_fail () from /usr/lib/system/
libsystem_c.dylib
#5  0x00007fff887109dc in __chk_fail_overflow ()
   from /usr/lib/system/libsystem_c.dylib
#6  0x00007fff88710ef8 in __sprintf_chk ()
   from /usr/lib/system/libsystem_c.dylib
#7  0x0000000100000ecc in main () at concat.c:8
```

Esercizio (Debugging)

- *Si esegua il debug del seguente file Concatena.c per capire quale problema comporta al suo interno*

```
(gdb) c
Continuing.
Program terminated with signal SIGABRT, Aborted.
The program no longer exists.
(gdb) b main
Breakpoint 1 at 0x100000e9f: file concat.c, line 7.
(gdb) run
Starting program: /Users/rdevirgilio/Desktop/concat
Breakpoint 1, main () at concat.c:7
7        strcpy(s1,  "This is source”);
(gdb) n
8        sprintf(s2, "%s", "This is destination”);
(gdb) n
Program received signal SIGABRT, Aborted.
0x00007fff874dcf06 in ?? () from /usr/lib/system/libsystem_kernel.dylib
```

Esercizio (Debugging)

- *Si esegua il debug del seguente file fibonacci.c per capire quale problema comporta al suo interno*

```
#include <stdio.h>

#include <stdlib.h>


// Recursive_version

unsigned int fib(unsigned int n) {

    if (n < 2) return n;

    else return fib(n - 1) + fib(n - 2);

}
```

Esercizio (Debugging)

- *Si esegua il debug del seguente file fibonacci.c per capire quale problema comporta al suo interno*

```
// Iterative_version
unsigned int fib_iter(unsigned int n) {
    unsigned int i = 0, j = 1, k, t;
    for (k = 1; k <= n; ++k) {
        t = i + j;
        i = j;
        j = t;
    }
    return j;
}
```

Esercizio (Debugging)

- *Si esegua il debug del seguente file fibonacci.c per capire quale problema comporta al suo interno*

```
int main(int argc, char* argv[]) {
    if (argc < 2) {
    printf("Syntax: %s <n>\n", argv[0]);
        return 1;
    }
    unsigned int n = atoi(argv[1]);
    printf("[%u] ric: %u iter: %u\n", n, fib(n), fib_iter(n));
    return 0;
}
```

Esercizio (Debugging)

- *I due metodi fib e fib_iter sono stati presi da Wikibooks ma restituiscono risultati difformi. Risalire alla causa di questa difformità con l'ausilio di gdb, in particolare utilizzando breakpoint (possibilmente condizionali) per ispezionare il comportamento di fib_iter*

Esercizio (Debugging)

- *Promemoria su comandi utili di gdb*

```
(gdb) file <nome_eseguibile> // per caricare un eseguibile

(gdb) run [<arg1> ...] // per lanciare l'esecuzione con eventuali argomenti

(gdb) quit // per uscire da gdb (o CTRL-d)

(gdb) break mio_file.c:20 // inserisco un breakpoint alla linea 20 di mio_file.c

(gdb) break mio_file.c:20 if x == 0 // breakpoint condizionale

(gdb) info break // per mostrare breakpoint attivi

(gdb) cont // per riprendere l'esecuzione normalmente dopo un breakpoint

(gdb) step // prosegue l'esecuzione di una singola istruzione

(gdb) print x // stampa il contenuto della variabile x nello stack frame corrente

(gdb) clear mio_file.c:20 // per eliminare breakpoint su una locazione di
riferimento

(gdb) delete 1 // per eliminare il breakpoint contrassegnato come 1 da 'info
break'
```

Esercizio (Debugging)

- *Eseguendo il programma con input pari a `10` (scelto a caso), otteniamo il seguente output:*

```
$ gcc -Wall -g fibonacci.c -o fibonacci

$ ./fibonacci 10

[10] ric: 55 iter: 89
```

Esercizio (Debugging)

- *Possiamo notare come i risultati delle due varianti di fibonacci non siano uguali. Assumendo la correttezza della variante ricorsiva, non ci rimane che analizzare il comportamento della variante iterativa:*

```
11. unsigned int fib_iter(unsigned int n) {
12.     unsigned int i = 0, j = 1, k, t;
13.     for (k = 1; k <= n; ++k) {
14.         t = i + j;
15.         i = j;
16.         j = t; // al termine del corpo del for, j assume il valore di t
17.     }
18.     return j; // per n > 0, j sarà uguale a t calcolato all'ultima iterazione del for
19. }                    // se n == 1: t dopo un'iterazione
                    // se n == 2: t dopo due iterazioni
                    // e così via
```

Esercizio (Debugging)

- *Possiamo inserire un breakpoint all'ultima istruzione del for ed osservare come cambia la variabile t durante le varie iterazioni*
- *Sfortunatamente, settando il breakpoint sulla riga 17, gdb bloccherà l'esecuzione all'uscita dal for e non alla fine di ogni iterazione.*
- *Per tale motivo, siamo constretti a settare il breakpoint alla riga 16. Tuttavia, occorre ricordarsi che gdb bloccherà l'esecuzione prima di eseguire la linea di codice sui cui è settato un breakpoint e quindi esattamente prima dell'assegnamento della variabile j.*

```
11. unsigned int fib_iter(unsigned int n) {
12.     unsigned int i = 0, j = 1, k, t;
13.     for (k = 1; k <= n; ++k) {
14.         t = i + j;
15.         i = j;
16.         j = t; // al termine del corpo del for, j assume il valore di t
17.     }
18.     return j; // per n > 0, j sarà uguale a t calcolato all'ultima iterazione del for
19. }                 // se n == 1: t dopo un'iterazione
                  // se n == 2: t dopo due iterazioni
                  // e così via
```

Esercizio (Debugging)

```
$ gdb fibonacci

(gdb) break fibonacci.c:16

Breakpoint 1 at 0x4005e8: file fibonacci.c, line 16.

(gdb) run 10

Starting program: /media/sf_SC1-1617/lab-01/fibonacci 10

Breakpoint 1, fib_iter (n=10) at fibonacci.c:16

16          j = t;

(gdb) print t

$1 = 1

(gdb) continue

Continuing.

Breakpoint 1, fib_iter (n=10) at fibonacci.c:16

16          j = t;
```

Esercizio (Debugging)

Viene incontrato per la prima volta il breakpoint nell'esecuzione. Stampiamo il valore di t al fine di valutare il risultato della funzione per n == 1. Poi preseguiamo nell'esecuzione:

```
(gdb) print t

$1 = 1

(gdb) continue

Continuing.

Breakpoint 1, fib_iter (n=10) at fibonacci.c:16

16              j = t;
```

Esercizio (Debugging)

Viene incontrato per la seconda volta il breakpoint nell'esecuzione. Stampiamo il valore di t al fine di valutare il risultato della funzione per n == 2. Andiamo avanti nell'esecuzione, ripetendo la stampa del valore t ad ogni iterazione:

```
(gdb) print t

$2 = 2

(gdb) continue

Continuing.

Breakpoint 1, fib_iter (n=10) at fibonacci.c:16

16          j = t;

(gdb) print t

$3 = 3

(gdb) continue

Continuing.

Breakpoint 1, fib_iter (n=10) at fibonacci.c:16

16          j = t;
```

Esercizio (Debugging)

```
(gdb) print t

$4 = 5

(gdb) continue

Continuing.

Breakpoint 1, fib_iter (n=10) at fibonacci.c:16

16          j = t;

(gdb) print t

$5 = 8

(gdb) continue

Continuing.

Breakpoint 1, fib_iter (n=10) at fibonacci.c:16

16          j = t;

(gdb) print t

$6 = 13

(gdb) continue

Continuing.
```

Esercizio (Debugging)

```
(gdb) print t

$9 = 55

(gdb) continue

Continuing.

Breakpoint 1, fib_iter (n=10) at fibonacci.c:16

16              j = t;

(gdb) print t

$10 = 89

(gdb) continue

Continuing.

[10] ric: 55 iter: 89

[Inferior 1 (process 10077) exited normally]
```

Esercizio (Debugging)

Possiamo notare che lungo le varie iterazioni, t assume i seguenti valori {1, 2, 3, 5, 8, 13, 21, 34, 55, 89}.

Il valore 55 viene effettivamente calcolato anche dalla variante iterativa, ma una condizione del loop non corretta, porta il codice ad eseguire un'ulteriore, e non dovuta, iterazione. Analizzando il codice, emerge che la condizione del loop:

```
for (k = 1; k <= n; ++k) {
```

Andrebbe sostituita con:

```
for (k = 1; k < n; ++k) {
```

Esercizio (Debugging)

Fatta tale modifica, il risultato della versione iterativa per un input pari a 10 risulta uguale a quello della variante ricorsiva.

Si noti che durante la sessione di debugging, avremmo potuto analizzare le prime iterazioni, successivamente rimuovere il breakpoint inserito

del 1 dove l'ID 1 è stato ottenuto con info break

inserire un brekpoint condizionale

break fibonacci.c:16 if k >= n - 2

e proseguire la nostra analisi solo sulle ultime due iterazioni.

In tal modo, il numero di iterazione da analizzare si sarebbe ridotto notevolmente e probabilmente avrebbe comunque messo in luce il problema.

Esercizio (Debugging)

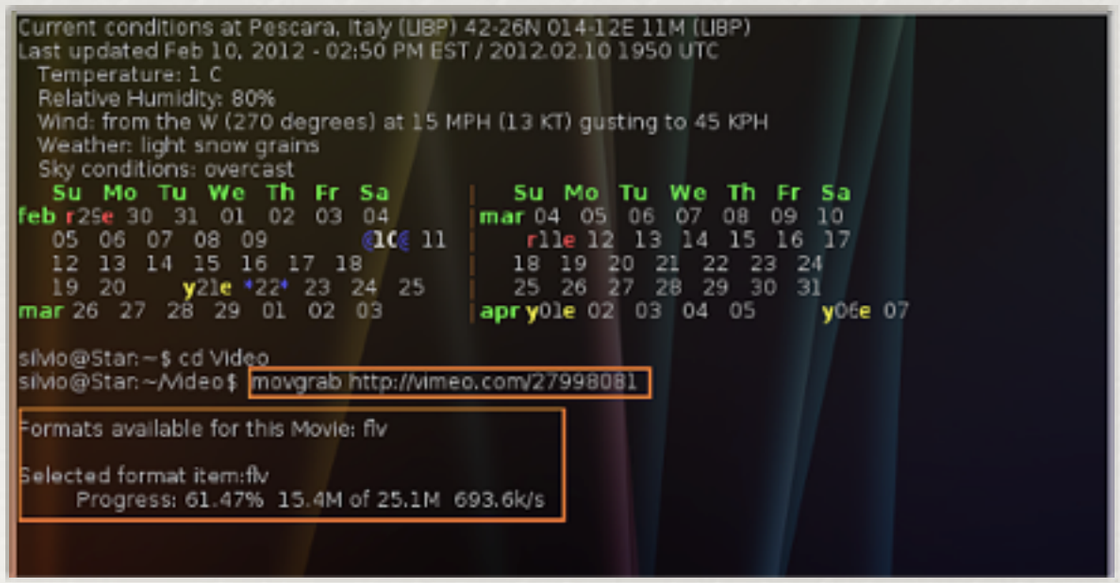
La variante iterativa, dopo la nostra modifica, continua a non essere equivalente alla variante ricorsiva.

Infatti, per un input pari a 0, la versione iterativa continua a ritornare un risultato non corretto.

Per tale motivo, occorre modificare ulteriormente la variante iterativa al fine di gestire questo caso di bordo.


Lesson
Learned!


LINUX
Professional Operating System
THANK YOU


Current conditions at Pescara, Italy (UBP) 42-26N 014-12E 11M (UBP)
Last updated Feb 10, 2012 - 02:50 PM EST / 2012.02.10 1950 UTC
Temperature: 1 C
Relative Humidity: 80%
Wind: from the W (270 degrees) at 15 MPH (13 KT) gusting to 45 KPH
Weather: light snow grains
Sky conditions: overcast
Su Mo Tu We Th Fr Sa
feb 26 27 28 29 01 02 03
05 06 07 08 09 10 11
12 13 14 15 16 17 18
19 20 21 22 23 24 25
mar 26 27 28 29 01 02 03
Su Mo Tu We Th Fr Sa
mar 04 05 06 07 08 09 10
11 12 13 14 15 16 17
18 19 20 21 22 23 24
25 26 27 28 29 30 31
apr 01 02 03 04 05 06 07
silvio@Star:~$ cd Video
silvio@Star:~/Video$ movgrab http://vimeo.com/27998081
Formats available for this Movie: flv
Selected format item:flv
Progress: 61.47% 15.4M of 25.1M 693.6k/s


LINUX
PCLinuxOS
Freedom. Choices. Beautiful.