UNIVERSITÀ DEGLI STUDI DI PALERMO

SISTEMI OPERATIVI

ING. MARCO MORANA

marco.morana@unipa.it

UNIVERSITÀ DEGLI STUDI DI PALERMO

Le comunicazioni o le registrazioni audio/video trasmesse o caricate su questa piattaforma dal Prof. Marco Morana sono effettuate al fine esclusivo di garantire l'erogazione della didattica in modalità telematica, in ossequio alle disposizione del D.P.C.M. del 4 marzo 2020 e della Circolare M.I.U.R. del 5 marzo 2020 correlate all'emergenza sanitaria legata alla diffusione del Covid-19.

Il consenso alla trasmissione/comunicazione di file audio/video contenenti dati personali di qualsivoglia natura (ex art 4, n. 1 Reg. UE 679/2016), riferiti alla propria persona, nonché contenuti/materiali didattici tutelati dalla Legge sul Diritto d'Autore, viene prestato dal sottoscritto, Prof. Marco Morana, a favore dell'Università degli Studi di Palermo nei limiti della liberatoria all'uopo sottoscritta.

Tale consenso è prestato esclusivamente allo scopo di consentire una fruizione del materiale audio/video di cui sopra, circoscritta ai soli studenti dell'Università degli Studi di Palermo che risultino registrati al Team.

UNIVERSITÀ DEGLI STUDI DI PALERMO

Nessun consenso è prestato dal Prof. Marco Morana alla circolazione di immagini, video o file audio , nonché contenuti /materiali didattici tutelati dalla Legge sul Diritto d'Autore, estratti o comunque ricavati dai contenuti caricati o trasmessi sulla piattaforma Microsoft Teams, fuori dai casi previsti dalla liberatoria resa all'Ateneo.
Qualunque violazione delle condizioni e dei limiti che accompagnano il consenso come sopra prestato e dipendenti da condotte degli studenti che risultino registrati al Team "(materia)" sarà da considerare illecita, ai sensi delle disposizioni normative vigenti ed esporrà alle conseguenze di legge.

Il sottoscritto, Prof. Marco Morana si riserva di agire in ogni sede a tutela dei propri "dati personali", della propria immagine e dei diritti d'autore, laddove in qualsiasi forma violati.

Thread programming

Threads Concepts

- Threads, like processes, allows a program to do more than one thing at a time
- As with processes, threads appear to run concurrently; the Linux kernel schedules them asynchronously, interrupting each thread to give others a chance to execute
- Conceptually, a thread exists within a process
- When you invoke a program, Linux creates a new process containing a single thread, which runs the program sequentially
- All threads run the same program in the same process, but each thread may be executing a different part of the program at any given time.

Process VS Thread

- Using fork, the child process is initially running its parent's program, with its parent's virtual memory, file descriptors, and so on copied.
  - The child process can modify its memory without affecting its parent, and vice versa.

- **The creating and the created thread share the same memory space, file descriptors, and other system resources as the original**.
  - If one thread changes the value of a variable, the other thread subsequently will see the modified value.
  - If one thread closes a file descriptor, other threads may not read from or write to that file descriptor.
  - …

P-threads and Linux

- GNU/Linux implements the POSIX standard thread API (known as pthreads – p(osix)threads)
- All thread functions and data types are declared in the header file **<pthread.h>**
- The pthread functions are not included in the standard C library, instead, they are in **libpthread**
- You should add **-lpthread** to the command line when you link your program

Thread creation

- Each thread in a process is identified by a *thread ID*.
- In C/C++ the type **pthread_t** is used to refer to thread IDs
- Upon creation, each thread executes a *thread function* containing the code that the thread should run
- When the *thread function* returns, the thread exits
- On GNU/Linux, thread functions take a single generic pointer `void*` parameter called *thread argument*, and have a `void*` return type
- Programs use:
  - *thread argument* to pass data to a new thread
  - the *return value* to pass data from an exiting thread back to its creator

Thread creation

```
#include <pthread.h>

int pthread_create(pthread_t *thread,
                   const pthread_attr_t *attr,
                   void *(*start_routine) (void *),
                   void *arg);
```

Thread creation

```
#include <pthread.h>

int pthread_create(pthread_t *thread,
                   const pthread_attr_t *attr,
                   void *(*start_routine) (void *),
                   void *arg);
```

A pointer to a pthread_t variable, in which the ID of the new thread is stored

Thread creation

```
#include <pthread.h>

int pthread_create(pthread_t *thread,
                   const pthread_attr_t *attr,
                   void *(*start_routine) (void *),
                   void *arg);
```

A pointer to a thread attribute object that controls details of how the thread interacts with the rest of the program (NULL = default attributes)

Thread creation

```
#include <pthread.h>

int pthread_create(pthread_t *thread,
                   const pthread_attr_t *attr,
                   void *(*start_routine) (void *),
                   void *arg);
```

A pointer to the thread function

Thread creation

```
#include <pthread.h>

int pthread_create(pthread_t *thread,
                   const pthread_attr_t *attr,
                   void *(*start_routine) (void *),
                   void *arg);
```

The argument passed to the thread function when the thread begins executing

Thread creation

```
#include <pthread.h>

int pthread_create(pthread_t *thread,
                   const pthread_attr_t *attr,
                   void *(*start_routine) (void *),
                   void *arg);
```

- A call to `pthread_create` returns immediately, and the original thread continues executing the instructions following the call.
- Meanwhile, the new thread begins executing the thread function.
- Linux schedules both threads asynchronously, and your program must not rely on the relative order in which instructions are executed in the two threads.

thread-create.c

```c
#include <pthread.h>
#include <stdio.h>
#include <unistd.h>

/* Prints -'s to stderr.  The parameter is unused.  Does not return.  */
void* print_X (void* unused) {
  while (1) {
    printf("-\n");
    fflush(stdout); // stdout stream is buffered, so we need
                    // to force it for showing its content

    sleep(1);       // sleep in seconds
  }
  return NULL;
}

/* The main program.  */
int main () {
  pthread_t thread_id;
  // thread id, NULL (default attributes), thrd function, args della thrd function
  pthread_create (&thread_id, NULL, &print_X, NULL);

  /* Print o's continuously */
  while (1) {
    printf("o\n");
    fflush(stdout);
    sleep(1);
  }
  return 0;
}
```

gcc -o thread-create thread-create.c -lpthread

ubuntu1804 [Running]

Attività Terminale ven 16:32

Apri thread-create.c ~/Scrivania/thread Salva

```c
#include <pthread.h>
#include <stdio.h>
#include <unistd.h>

/* Prints -'s to stderr.  The parameter is unused.  Does not return.  */
void* print_X (void* unused) {
  while (1) {
    printf("-\n");
    fflush(stdout); // stdout stream is buffered, so we need
                    // to force it for showing its content

    sleep(1);       // sleep in seconds
  }
  return NULL;
}

/* The main program.  */
int main () {

  pthread_t thread_id;

  // thread id - NULL (default attributes) - th funct
  pthread_create(&thread_id, NULL, &print_X, NULL);

  /* Print o's continuously */
  while (1) {
    printf("o\n");
    fflush(stdout);
    sleep(1);
  }
  return 0;

}
```

marco@marco-VirtualBox: ~/Scrivania/thread

File Modifica Visualizza Cerca Terminale Aiuto

```
marco@marco-VirtualBox:~/Scrivania/thread$ gcc -o thread-create thread-create.c
 -lpthread
marco@marco-VirtualBox:~/Scrivania/thread$ ./thread-create
o
-
o
-
o
-
o
-
o
-
o
-
o
-
o
-
```

Left ⌘

gcc -o thread-create thread-create.c -lpthread

ubuntu1804 [Running]

Attività Terminale ven 16:50

Apri thread-create.c ~/Scrivania/thread Salva

```c
#include <pthread.h>
#include <stdio.h>
#include <unistd.h>

/* Prints -'s to stderr.  The parameter is unused.  Does not return.  */
void* print_X (void* unused) {
  while (1) {
    printf("-\n");
    fflush(stdout); // stdout stream is buffered, so we need
                    // to force it for showing its content

    sleep(2);       // sleep in seconds
  }
  return NULL;
}

/* The main program.  */
int main () {

  pthread_t thread_id;

  // thread id - NULL (default attributes) - th funct
  pthread_create(&thread_id, NULL, &print_X, NULL);

  /* Print o's continuously */
  while (1) {
    printf("o\n");
    fflush(stdout);
    sleep(1);
  }
  return 0;

}
```

marco@marco-VirtualBox: ~/Scrivania/thread

File Modifica Visualizza Cerca Terminale Aiuto

```
marco@marco-VirtualBox:~/Scrivania/thread$ gcc -o thread-create thread-create.c
 -lpthread
marco@marco-VirtualBox:~/Scrivania/thread$ ./thread-create
o
-
o
o
-
o
o
-
o
-
o
o
o
-
o
```

Left ⌘

See running threads

```
ps -e -o pid,cmd
```

```
marco@marco-VirtualBox: ~
File  Modifica  Visualizza  Cerca  Terminale  Aiuto
 3020 [jfsSync]
 3466 /usr/lib/gvfs/gvfsd-network --spawner :1.13 /org/gtk/gvfs/exec_spaw/1
 3488 /usr/lib/gvfs/gvfsd-smb-browse --spawner :1.13 /org/gtk/gvfs/exec_spaw/6
 3501 /usr/lib/gvfs/gvfsd-dnssd --spawner :1.13 /org/gtk/gvfs/exec_spaw/7
 3909 [kworker/0:2]
 7253 /usr/lib/gnome-terminal/gnome-terminal-server
 7263 bash
 9692 [kworker/u2:1]
 9968 [kworker/u2:0]
10017 /sbin/dhclient -d -q -sf /usr
10171 ./thread-create
10184 bash
10268 [kworker/u2:4]
10269 /usr/sbin/cupsd -l
```

```
top -H -p pid
```

```
marco@marco-VirtualBox: ~
File  Modifica  Visualizza  Cerca  Terminale  Aiuto
top - 11:20:30 up  4:42,  1 user,  load average: 0,01, 0,03, 0,02
Threads:   2 total,   0 running,   2 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0,0 us,  0,0 sy,  0,0 ni,100,0 id,  0,0 wa,  0,0 hi,  0,0 si,  0,0 st
KiB Mem :  1009088 total,    76544 free,   573696 used,   358848 buff/cache
KiB Swap:   483800 total,    28724 free,   455076 used.   213856 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S %CPU %MEM     TIME+ COMMAND
10171 marco     20   0   14888    936    856 S  0,0  0,1   0:00.03 thread-crea+
10172 marco     20   0   14888    936    856 S  0,0  0,1   0:00.01 thread-crea+
```

```
marco@marco-VirtualBox: ~
File  Modifica  Visualizza  Cerca  Terminale  Aiu
marco@marco-VirtualBox:~$ ps -T -p 101
  PID  SPID TTY          TIME CMD
10171 10171 pts/0    00:00:00 thread-c
10171 10172 pts/0    00:00:00 thread-c
marco@marco-VirtualBox:~$
```

2 thread, di cui uno è il main

```
ps -T -p pid
```

Passing Data to Threads

```
int pthread_create(pthread_t *thread,
                   const pthread_attr_t *attr,
                   void *(*start_routine) (void *),
                   void *arg);
```

- Because the type of the argument is `void*`, this element is used to pass a pointer to some structure or array of data
- One common technique is to define a structure for each thread function, which contains the “parameters” that the thread function expects

thread-create2.c

```c
#include <pthread.h>
#include <stdio.h>
#include <unistd.h>

struct print_params {
  char character;
  int times;
};

int main () {
  pthread_t thread1_id;
  pthread_t thread2_id;

  struct print_params thread1_args;
  struct print_params thread2_args;

  thread1_args.character = 'X';
  thread1_args.times = 20;

  thread2_args.character = 'o';
  thread2_args.times = 30;

  pthread_create (&thread1_id, NULL, &print_char, &thread1_args);
  pthread_create (&thread2_id, NULL, &print_char, &thread2_args);




  return 0;
}
```

```c
void* print_char (void* parameters)
{
  struct print_params* pp = (struct
print_params*) parameters;

  for (int i=0; i<= pp->times; i++) {
    printf("%c",pp->character);
    fflush(stdout);
    sleep(1);
  }
  return NULL;
}
```

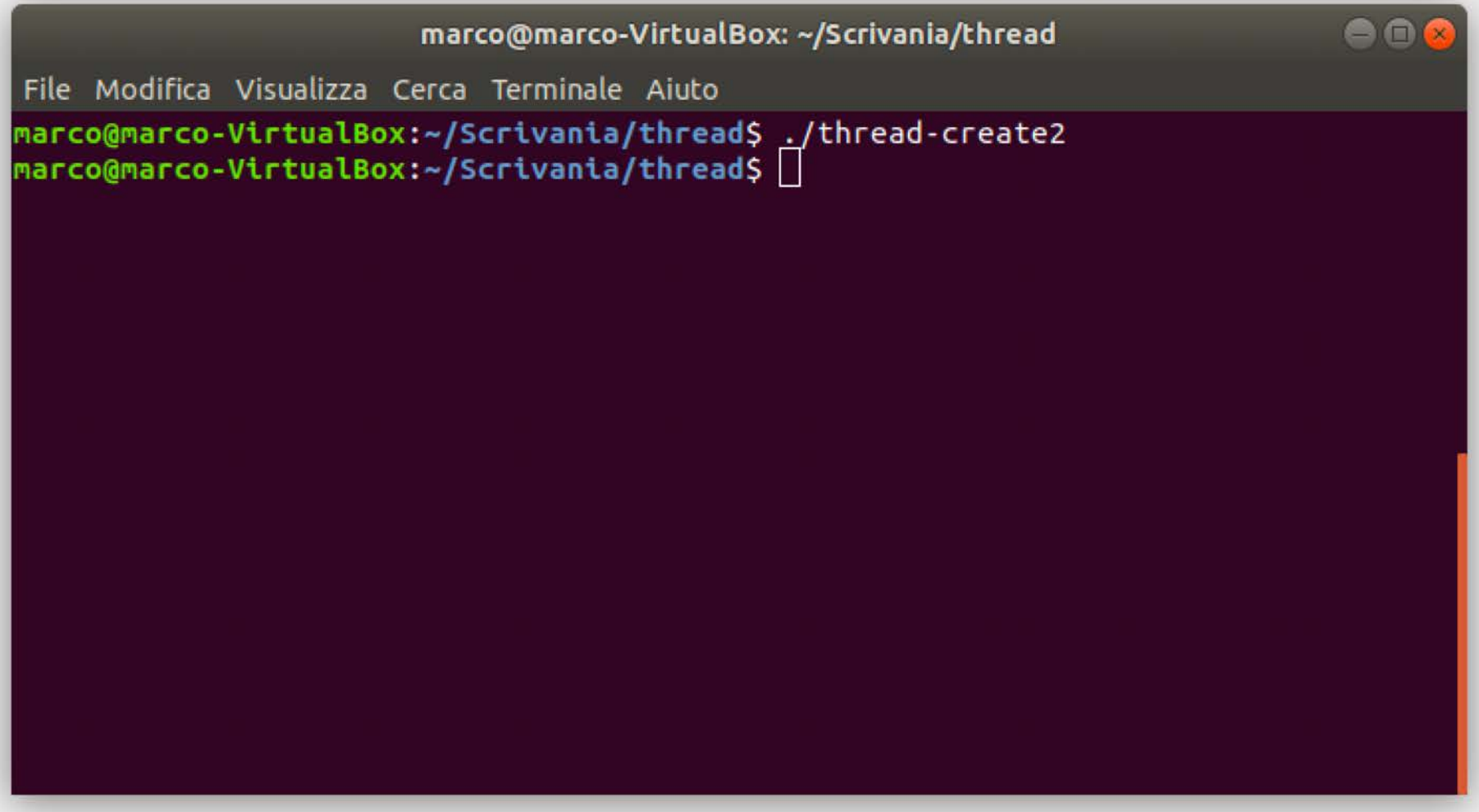
thread-create2.c

```c
#include <pthread.h>
#include <stdio.h>
#include <unistd.h>

struct print_params {
  char character;
  int times;
};

int main () {
  pthread_t thread1_id;
  pthread_t thread2_id;

  struct print_params thread1_args;
  struct print_params thread2_args;

  thread1_args.character = 'X';
  thread1_args.times = 20;

  thread2_args.character = 'o';
  thread2_args.times = 30;

  pthread_create (&thread1_id, NULL, &print_char, &thread1_args);
  pthread_create (&thread2_id, NULL, &print_char, &thread2_args);



  return 0;
}
```

```c
void* print_char (void* parameters)
{
  struct print_params* pp = (struct
print_params*) parameters;

  for (int i=0; i<= pp->times; i++) {
    printf("%c",pp->character);
    fflush(stdout);
    sleep(1);
  }
  return NULL;
}
```

Cosa succede ai due thread se nel frattempo
il main viene schedulato, eseguito e terminato?

thread-create2.c



Joining Threads

- Prevent Linux from scheduling threads in such a way that main finishes executing before the other threads are done

```
#include <pthread.h>
int pthread_join(pthread_t thread, void **value_ptr);
```

the thread ID of the thread to wait for

a pointer to a void* variable that will receive the finished thread's return value

thread-create2.c

```c
#include <pthread.h>
#include <stdio.h>
#include <unistd.h>

struct print_params {
  char character;
  int times;
};

int main () {
  pthread_t thread1_id;
  pthread_t thread2_id;

  struct print_params thread1_args;
  struct print_params thread2_args;

  thread1_args.character = 'X';
  thread1_args.times = 20;

  thread2_args.character = 'o';
  thread2_args.times = 30;

  pthread_create (&thread1_id, NULL, &print_char, &thread1_args);
  pthread_create (&thread2_id, NULL, &print_char, &thread2_args);

  pthread_join(thread1_id,NULL);
  pthread_join(thread2_id,NULL);

  return 0;
}
```

```c
void* print_char (void* parameters)
{
  struct print_params* pp = (struct
print_params*) parameters;

  for (int i=0; i<= pp->times; i++) {
    printf("%c",pp->character);
    fflush(stdout);
    sleep(1);
  }
  return NULL;
}
```

Il secondo argomento è un puntatore al valore di ritorno del thread, se non esiste si usa NULL

ubuntu1804 [Running]

Attività Terminale ven 17:06

Apri thread-create2.c ~/Scrivania/thread Salva

```c
struct print_params {
  char character;
  int times;
};

void* print_char (void* parameters) {
  struct print_params* pp = (struct print_params*) parameters;

  for (int i=0; i<= pp->times; i++) {
    printf("%c",pp->character);
    fflush(stdout);
    sleep(1);
  }
  return NULL;
}

int main () {
  pthread_t thread1_id;
  pthread_t thread2_id;

  struct print_params thread1_args;
  struct print_params thread2_args;

  thread1_args.character = 'X';
  thread1_args.times = 20;
  thread2_args.character = 'o';
  thread2_args.times = 30;

  pthread_create (&thread1_id, NULL, &print_char,&thr
  pthread_create (&thread2_id, NULL, &print_char,&thr

  pthread_join(thread1_id,NULL);
  pthread_join(thread2_id,NULL);

  return 0;
}
```

marco@marco-VirtualBox: ~/Scrivania/thread

File Modifica Visualizza Cerca Terminale Aiuto

```
marco@marco-VirtualBox:~/Scrivania/thread$ ./thread-create2
oXoXoXoXoXoXoXoXoXoXoXoXoXoXoXoXoXoXoXoXXooXoooooooo
```

Left ⌘

See running threads

```
marco@marco-VirtualBox: ~
File  Modifica  Visualizza  Cerca  Terminale  Aiuto
top - 11:26:25 up  4:48,  1 user,  load average: 0,03, 0,01, 0,00
Threads:   3 total,   0 running,   3 sleeping,   0 stopped,   0 zombie
%Cpu(s):  1,3 us,  0,0 sy,  0,0 ni, 98,7 id,  0,0 wa,  0,0 hi,  0,0 si,  0,0 st
KiB Mem :  1009088 total,    73788 free,   575524 used,   359776 buff/cache
KiB Swap:   483800 total,    28724 free,   455076 used.   211960 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S %CPU %MEM     TIME+ COMMAND
10463 marco     20   0   88620    812    732 S  0,0  0,1   0:00.00 thread-crea+
10464 marco     20   0   88620    812    732 S  0,0  0,1   0:00.00 thread-crea+
10465 marco     20   0   88620    812    732 S  0,0  0,1   0:00.00 thread-crea+
```

3 thread, di cui uno è il main

Thread Return Values

- If the second argument you pass to `pthread_join` is non-null, the thread's return value will be placed in the location pointed to by that argument

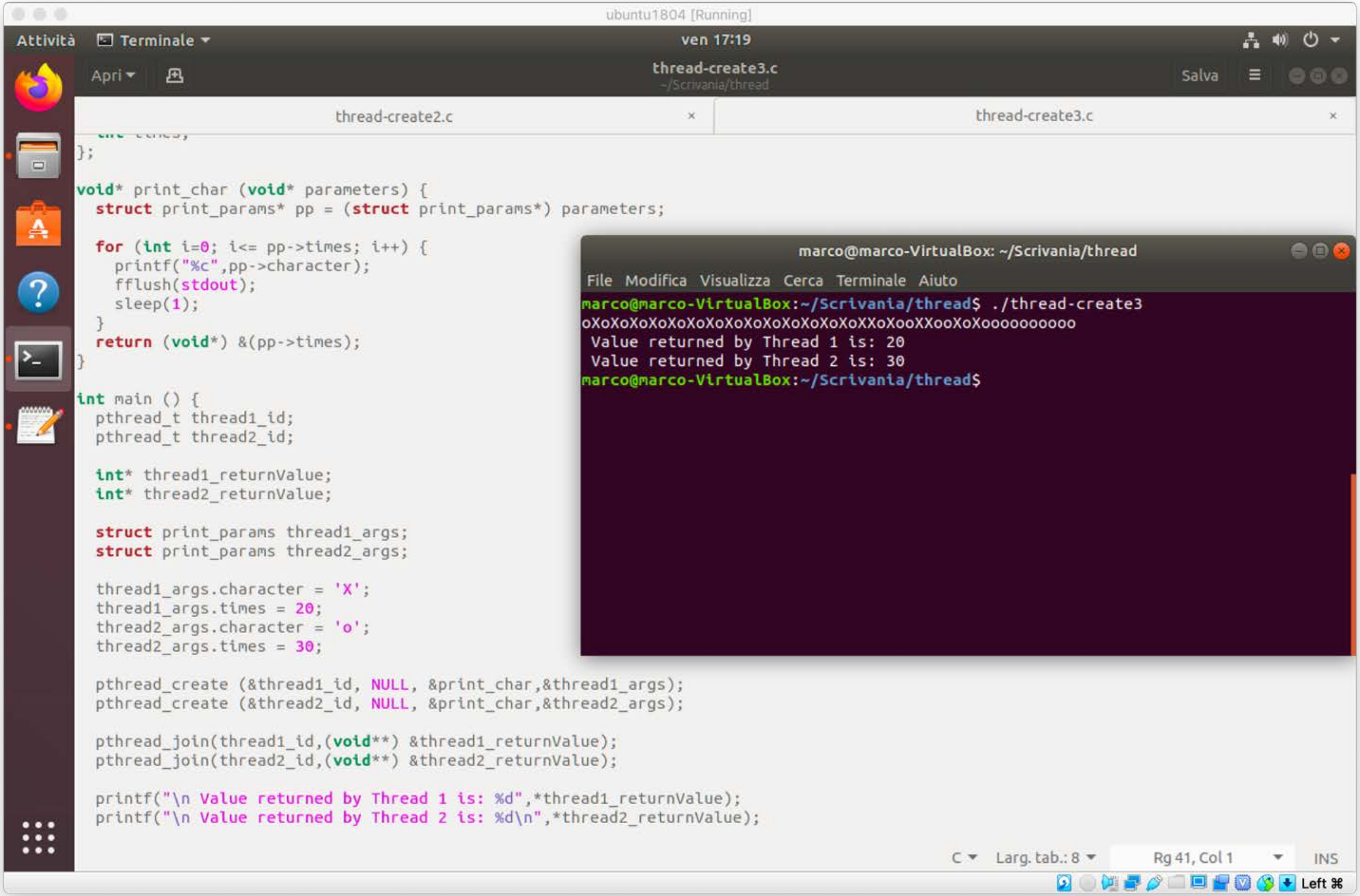
- The thread return value, like the thread argument, is of type void*

- If you want to pass back a single int or other small number, you can do this by *casting the value to void* and then casting back* to the appropriate type after calling `pthread_join`

thread-create3.c

```c
#include <pthread.h>
#include <stdio.h>
#include <unistd.h>

struct print_params {
  char character;
  int times;
};

int main () {
  pthread_t thread1_id;
  pthread_t thread2_id;

  int* thread1_returnValue;
  int* thread2_returnValue;

  struct print_params thread1_args;
  struct print_params thread2_args;

  thread1_args.character = 'X';
  thread1_args.times = 20;
  thread2_args.character = 'o';
  thread2_args.times = 30;

  pthread_create (&thread1_id, NULL, &print_char, &thread1_args);
  pthread_create (&thread2_id, NULL, &print_char, &thread2_args);

  pthread_join(thread1_id, (void**) &thread1_returnValue);
  pthread_join(thread2_id, (void**) &thread2_returnValue);

  printf("\n Value returned by Thread 1 is: %d",*thread1_returnValue);
  printf("\n Value returned by Thread 2 is: %d\n",*thread2_returnValue);
  return 0;
}
```

```c
void* print_char (void* parameters)
{
  struct print_params* pp = (struct
print_params*) parameters;

  for (int i=0; i<= pp->times; i++) {
    printf("%c",pp->character);
    fflush(stdout);
    sleep(1);
  }
  return (void*) &(pp->times);
}
```



```c
};

void* print_char (void* parameters) {
  struct print_params* pp = (struct print_params*) parameters;

  for (int i=0; i<= pp->times; i++) {
    printf("%c",pp->character);
    fflush(stdout);
    sleep(1);
  }
  return (void*) &(pp->times);
}

int main () {
  pthread_t thread1_id;
  pthread_t thread2_id;

  int* thread1_returnValue;
  int* thread2_returnValue;

  struct print_params thread1_args;
  struct print_params thread2_args;

  thread1_args.character = 'X';
  thread1_args.times = 20;
  thread2_args.character = 'o';
  thread2_args.times = 30;

  pthread_create (&thread1_id, NULL, &print_char,&thread1_args);
  pthread_create (&thread2_id, NULL, &print_char,&thread2_args);

  pthread_join(thread1_id,(void**) &thread1_returnValue);
  pthread_join(thread2_id,(void**) &thread2_returnValue);

  printf("\n Value returned by Thread 1 is: %d",*thread1_returnValue);
  printf("\n Value returned by Thread 2 is: %d\n",*thread2_returnValue);
```

```
marco@marco-VirtualBox:~/Scrivania/thread$ ./thread-create3
oXoXoXoXoXoXoXoXoXoXoXoXoXoXoXXoXooXXooXoXoooooooooo
 Value returned by Thread 1 is: 20
 Value returned by Thread 2 is: 30
marco@marco-VirtualBox:~/Scrivania/thread$
```

4.11 Java Multithreading Case Study, Part I: Introduction to Java Threads

- Java allows the application programmer to create threads that can port to many computing platforms
- Threads
  - Created by class `Thread`
  - Execute code specified in a `Runnable` object's `run` method
- Java supports operations such as naming, starting and joining threads

Thread creation in Java (`ThreadTester.java`)

```
public class ThreadTester {

        public static void main(String [] args) {

                //creiamo 3 thread
                PrintThread thread1 = new PrintThread("thread 1");
                PrintThread thread2 = new PrintThread("thread 2");
                PrintThread thread3 = new PrintThread("thread 3");

                System.err.println("Starting threads");

                // start, place the thread in ready state
                thread1.start();
                thread2.start();
                thread3.start();

                System.err.println("Threads started, main ends\n");

        }

}
```

Thread creation in Java (`ThreadTester.java`)

```
class PrintThread extends Thread {

        private int sleepTime;

        // costruttore
        public PrintThread(String name) {
                super(name);
                // sleep 0-5 secondi
                sleepTime = (int) (Math.random() * 5001);
        }

        public void run() {
                try {
                        System.err.println(getName() + " going to
                        sleep for " +  sleepTime + "milliseconds");
                        Thread.sleep(sleepTime);
                } catch (InterruptedException e) {
                        e.printStackTrace();
                }
                System.err.println(getName() + " done sleeping");
        }
}
```

ubuntu1804 [Running]

Attività Terminale ven 17:26

Apri ThreadTester.java ~/Scrivania/thread Salva

```java
        public static void main(String [] args) {

                //creiamo 3 thread
                PrintThread thread1 = new PrintThread("thread 1");
                PrintThread thread2 = new PrintThread("thread 2");
                PrintThread thread3 = new PrintThread("thread 3");

                System.err.println("Starting threads"

                // start, place the thread in ready s
                thread1.start();
                thread2.start();
                thread3.start();

                System.err.println("Threads started,
        }
}

class PrintThread extends Thread {
        private int sleepTime;

        // costruttore
        public PrintThread(String name) {
                super(name);
                // sleep 0-5 secondi
                sleepTime = (int) (Math.random() * 50
        }

        public void run() {
                try {
                        System.err.println(getName() + " going to sleep for " + sleepTime + " milliseconds");
                        Thread.sleep(sleepTime);
                } catch (InterruptedException e) {
                        e.printStackTrace();
                }

                System.err.println(getName() + " done sleeping");
        }
}
```

Java Larg. tab.: 8 Rg 33, Col 95 INS

marco@marco-VirtualBox: ~/Scrivania/thread

File Modifica Visualizza Cerca Terminale Aiuto

```
marco@marco-VirtualBox:~/Scrivania/thread$ javac ThreadTester.java
marco@marco-VirtualBox:~/Scrivania/thread$ java ThreadTester
Starting threads
Threads started, main ends

thread 3 going to sleep for 1158 milliseconds
thread 1 going to sleep for 3296 milliseconds
thread 2 going to sleep for 4124 milliseconds
thread 3 done sleeping
thread 1 done sleeping
thread 2 done sleeping
marco@marco-VirtualBox:~/Scrivania/thread$
```

Left ⌘

marco@marco-VirtualBox: ~

File Modifica Visualizza Cerca Terminale Aiuto

```
marco@marco-VirtualBox:~$ ps -T -p 11123
  PID  SPID TTY          TIME CMD
11123 11123 pts/0    00:00:00 java
11123 11124 pts/0    00:00:00 java
11123 11125 pts/0    00:00:00 VM Thread
11123 11126 pts/0    00:00:00 Reference Handl
11123 11127 pts/0    00:00:00 Finalizer
11123 11128 pts/0    00:00:00 Signal Dispatch
11123 11129 pts/0    00:00:00 C2 CompilerThre
11123 11130 pts/0    00:00:00 C1 CompilerThre
11123 11131 pts/0    00:00:00 Sweeper thread
11123 11132 pts/0    00:00:00 Service Thread
11123 11133 pts/0    00:00:00 Common-Cleaner
11123 11134 pts/0    00:00:00 VM Periodic Tas
11123 11135 pts/0    00:00:00 thread 1
11123 11136 pts/0    00:00:00 thread 2
11123 11137 pts/0    00:00:00 thread 3
marco@marco-VirtualBox:~$
```

marco@marco-VirtualBox: ~

File Modifica Visualizza Cerca Terminale Aiuto

```
top - 14:55:06 up  8:16,  1 user,  load average: 0,07, 0,08, 0,03
Threads:  15 total,   1 running,  14 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0,0 us,  0,0 sy,  0,0 ni, 99,7 id,  0,3 wa,  0,0 hi,  0,0 si,  0,0 st
KiB Mem :  1009088 total,    97476 free,   620296 used,   291316 buff/cache
KiB Swap:   483800 total,    48604 free,   435196 used.   184644 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S %CPU %MEM     TIME+ COMMAND
11162 marco     20   0 2354356  32996  18032 S  0,0  3,3   0:00.00 java
11163 marco     20   0 2354356  32996  18032 S  0,0  3,3   0:00.05 java
11164 marco     20   0 2354356  32996  18032 S  0,0  3,3   0:00.00 VM Thread
11165 marco     20   0 2354356  32996  18032 S  0,0  3,3   0:00.00 Reference H+
11166 marco     20   0 2354356  32996  18032 S  0,0  3,3   0:00.00 Finalizer
11167 marco     20   0 2354356  32996  18032 S  0,0  3,3   0:00.00 Signal Disp+
11168 marco     20   0 2354356  32996  18032 S  0,0  3,3   0:00.02 C2 Compiler+
11169 marco     20   0 2354356  32996  18032 S  0,0  3,3   0:00.03 C1 Compiler+
11170 marco     20   0 2354356  32996  18032 S  0,0  3,3   0:00.00 Sweeper thr+
11171 marco     20   0 2354356  32996  18032 S  0,0  3,3   0:00.00 Service Thr+
11172 marco     20   0 2354356  32996  18032 S  0,0  3,3   0:00.00 Common-Clea+
11173 marco     20   0 2354356  32996  18032 R  0,0  3,3   0:00.00 VM Periodic+
11174 marco     20   0 2354356  32996  18032 S  0,0  3,3   0:00.01 thread 1
11175 marco     20   0 2354356  32996  18032 S  0,0  3,3   0:00.01 thread 2
11176 marco     20   0 2354356  32996  18032 S  0,0  3,3   0:00.01 thread 3
```

jstack <pid>

Print Java stack traces of Java threads for a specified Java process

```
`Full thread dump Java HotSpot(TM) 64-Bit Server VM (10.0.1+10 mixed mode):

Threads class SMR info:
_java_thread_list=0x00007f1370007660, length=13, elements={
0x00007f13a008a800, 0x00007f13a008c800, 0x00007f13a00a0000, 0x00007f13a00a2000,
0x00007f13a00a4000, 0x00007f13a00a6000, 0x00007f13a0127000, 0x00007f13a00c7000,
0x00007f13a0142800, 0x00007f13a0144000, 0x00007f13a0145800, 0x00007f13a0011000,
0x00007f1370006000
}

"Reference Handler" #2 daemon prio=10 os_prio=0 tid=0x00007f13a008a800 nid=0x2c7d waiting on condition  [0x00007f13a4759000]
   java.lang.Thread.State: RUNNABLE
                        at java.lang.ref.Reference.waitForReferencePendingList(java.base@10.0.1/Native Method)
                        at java.lang.ref.Reference.processPendingReferences(java.base@10.0.1/Reference.java:174)
                        at java.lang.ref.Reference.access$000(java.base@10.0.1/Reference.java:44)
                        at java.lang.ref.Reference$ReferenceHandler.run(java.base@10.0.1/Reference.java:138)

"Finalizer" #3 daemon prio=8 os_prio=0 tid=0x00007f13a008c800 nid=0x2c7e in Object.wait()  [0x00007f13a4658000]
   java.lang.Thread.State: WAITING (on object monitor)
                        at java.lang.Object.wait(java.base@10.0.1/Native Method)
                        - waiting on <0x00000000f0809480> (a java.lang.ref.ReferenceQueue$Lock)
                        at java.lang.ref.ReferenceQueue.remove(java.base@10.0.1/ReferenceQueue.java:151)
                        - waiting to re-lock in wait() <0x00000000f0809480> (a java.lang.ref.ReferenceQueue$Lock)
                        at java.lang.ref.ReferenceQueue.remove(java.base@10.0.1/ReferenceQueue.java:172)
                        at java.lang.ref.Finalizer$FinalizerThread.run(java.base@10.0.1/Finalizer.java:216)

"Signal Dispatcher" #4 daemon prio=9 os_prio=0 tid=0x00007f13a00a0000 nid=0x2c7f runnable  [0x0000000000000000]
   java.lang.Thread.State: RUNNABLE

"C2 CompilerThread0" #5 daemon prio=9 os_prio=0 tid=0x00007f13a00a2000 nid=0x2c80 waiting on condition  [0x0000000000000000]
   java.lang.Thread.State: RUNNABLE
   No compile task

"C1 CompilerThread1" #6 daemon prio=9 os_prio=0 tid=0x00007f13a00a4000 nid=0x2c81 waiting on condition  [0x0000000000000000]
   java.lang.Thread.State: RUNNABLE
   No compile task

"Sweeper thread" #7 daemon prio=9 os_prio=0 tid=0x00007f13a00a6000 nid=0x2c82 runnable  [0x0000000000000000]
   java.lang.Thread.State: RUNNABLE

"Service Thread" #8 daemon prio=9 os_prio=0 tid=0x00007f13a0127000 nid=0x2c83 runnable  [0x0000000000000000]
   java.lang.Thread.State: RUNNABLE

"Common-Cleaner" #9 daemon prio=8 os_prio=0 tid=0x00007f13a00c7000 nid=0x2c84 in Object.wait()  [0x00007f138526a000]
   java.lang.Thread.State: TIMED_WAITING (on object monitor)
                        at java.lang.Object.wait(java.base@10.0.1/Native Method)
                        - waiting on <0x00000000f08d81d8> (a java.lang.ref.ReferenceQueue$Lock)
                        at java.lang.ref.ReferenceQueue.remove(java.base@10.0.1/ReferenceQueue.java:151)
                        - waiting to re-lock in wait() <0x00000000f08d81d8> (a java.lang.ref.ReferenceQueue$Lock)
                        at jdk.internal.ref.CleanerImpl.run(java.base@10.0.1/CleanerImpl.java:148)
                        at java.lang.Thread.run(java.base@10.0.1/Thread.java:844)
                        at jdk.internal.misc.InnocuousThread.run(java.base@10.0.1/InnocuousThread.java:134)

"thread 1" #10 prio=5 os_prio=0 tid=0x00007f13a0142800 nid=0x2c86 waiting on condition  [0x00007f1384c3f000]
   java.lang.Thread.State: TIMED_WAITING (sleeping)
                        at java.lang.Thread.sleep(java.base@10.0.1/Native Method)
                        at PrintThread.run(ThreadTester.java:34)

"thread 2" #11 prio=5 os_prio=0 tid=0x00007f13a0144000 nid=0x2c87 waiting on condition  [0x00007f1384b3e000]
   java.lang.Thread.State: TIMED_WAITING (sleeping)
                        at java.lang.Thread.sleep(java.base@10.0.1/Native Method)
                        at PrintThread.run(ThreadTester.java:34)

"thread 3" #12 prio=5 os_prio=0 tid=0x00007f13a0145800 nid=0x2c88 waiting on condition  [0x00007f1384a3d000]
   java.lang.Thread.State: TIMED_WAITING (sleeping)
                        at java.lang.Thread.sleep(java.base@10.0.1/Native Method)
                        at PrintThread.run(ThreadTester.java:34)

"DestroyJavaVM" #13 prio=5 os_prio=0 tid=0x00007f13a0011000 nid=0x2c7b waiting on condition  [0x0000000000000000]
   java.lang.Thread.State: RUNNABLE

"Attach Listener" #14 daemon prio=9 os_prio=0 tid=0x00007f1370006000 nid=0x2c99 runnable  [0x0000000000000000]
   java.lang.Thread.State: RUNNABLE

"VM Thread" os_prio=0 tid=0x00007f13a0080800 nid=0x2c7c runnable

"VM Periodic Task Thread" os_prio=0 tid=0x00007f13a00c8800 nid=0x2c85 waiting on condition

JNI global references: 3
```

Thread joining in Java (`Thread4Max.java`)

```
public class Thread4Max extends Thread {

  private int[] values;
  private int a,b;
  private int max;



  public static void main(String[] args) throws InterruptedException
  {

     int[] arr = new int[100];

     for (int i=0; i<arr.length; i++) {
          arr[i] = (int) (Math.random() * 1001); //oppure = i;
     }

     int max = getMax(arr);
     System.out.println("Massimo = " + max);
  }
```

Thread joining in Java (`Thread4Max.java`)

```
public static int getMax(int[] values) throws InterruptedException {
          int len = values.length;
          int max;

          // creiamo 2 thread
          Thread4Max thread1 = new Thread4Max(values, 0, len/2);
          System.out.println("Il thread 1 calcola il massimo dei valori in
                              posizione da 1 a " + len/2 );

          Thread4Max thread2 = new Thread4Max(values, len/2, len);
          System.out.println("Il thread 2 calcola il massimo dei valori in
                              posizione da " + len/2 + " a " + len );

          thread1.start();
          thread2.start();

}
```

Thread joining in Java (`Thread4Max.java`)

```
public Thread4Max(int[] values, int a, int b ) {
      this.a = a;
      this.b = b;
      this.values = values;
}

public void run() {
      max = values[a];

      for(int i=a+1; i<b; i++) {
        if(values[i]>max) {
          max = values[i];
        }
      }
}
```

Thread joining in Java (`Thread4Max.java`)

```
public static int getMax(int[] values) throws InterruptedException {
          int len = values.length;
          int max;

          // creiamo 2 thread
          Thread4Max thread1 = new Thread4Max(values, 0, len/2);
          System.out.println("Il thread 1 calcola il massimo dei valori in
                              posizione da 1 a " + len/2 );

          Thread4Max thread2 = new Thread4Max(values, len/2, len);
          System.out.println("Il thread 2 calcola il massimo dei valori in
                              posizione da " + len/2 + " a " + len );

          thread1.start();
          thread2.start();

          // aspettiamo il completamento e calcoliamo il max
          thread1.join();
          thread2.join();

          System.out.println("Massimo thread1 = " + thread1.max);
          System.out.println("Massimo thread2 = " + thread2.max);

          if(thread1.max > thread2.max) {
            max = thread1.max;
          } else {
            max = thread2.max;
          }
          return max;
}
```

Thread joining in Java (`Thread4Max.java`)

```
marco@marco-VirtualBox: ~/Scrivania/thread
File  Modifica  Visualizza  Cerca  Terminale  Aiuto
marco@marco-VirtualBox:~/Scrivania/thread$ java Thread4Max
Il thread 1 calcola il massimo dei valori in posizione da 1 a 50
Il thread 2 calcola il massimo dei valori in posizione da 50 a 100
Massimo thread1 = 987
Massimo thread2 = 977
Massimo = 987
marco@marco-VirtualBox:~/Scrivania/thread$
```