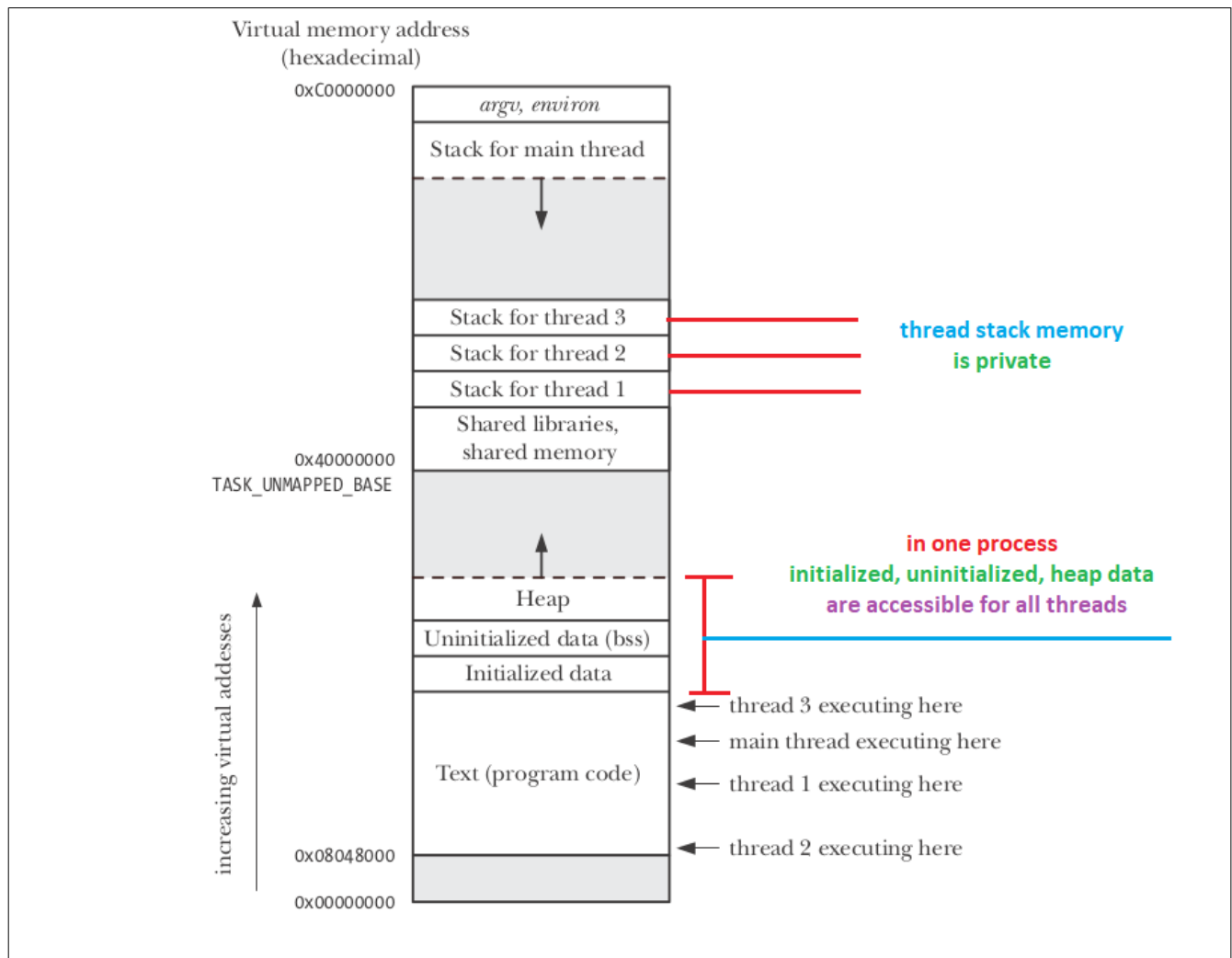


THREAD

Thread is a mechanism that permits an application to perform multiple tasks concurrently. A process can contain multiple threads.



Memory

Stack

Stack is a portion of memory (in RAM). It grows and shrinks similar to a stack data structure, things can be added and removed at the top.

When a thread is created, it is allocated a stack space. If we store more information than stack capacity, we will get a stack overflow and crash.

Default thread stack size for a few architectures

i386	2	MB
x86_64	2	MB
IA-64	32	MB

PowerPC	4 MB
---------	------

However, the stack size can be explicitly set in the attribute argument used to create thread by using `pthread_attr_setstacksize`.

Initialized Data, Uninitialized Data & Heap

In one process, threads all share the same global memory, including:

Initialized data	<i>global initialized variables</i> <i>global static initialized variables</i> <i>local static initialized variables</i>
Uninitialized data	<i>global uninitialized variables</i> <i>global static uninitialized variables</i> <i>local static uninitialized variables</i>
Heap data	<i>allocated by new, malloc, calloc</i>

Practise

Check thread default stack size

```
ulimit -s
```

```
invistd@server:~$ ulimit -s
8192
invistd@server:~$ _
```

Errno

In threaded programs, each thread has its own errno value.

Reason:

- To avoid race conditions

Example

```
#include <pthread.h>
#include <stdio.h>
#include <errno.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>

void* func_1() {
    errno = 1;
    printf("func_1, errno value: %d\n", errno);
}

void* func_2() {
```

```

        printf("func_2, errno value: %d\n", errno);
    }

int main() {
    pthread_t thread_1;
    pthread_create(&thread_1, NULL, func_1, NULL);
    sleep(1);

    pthread_t thread_2;
    pthread_create(&thread_2, NULL, func_2, NULL);
    sleep(1);

    return 0;
}

```

In above program,

- thread_1 set errno to 1. However, thread_1 set **its own errno** instead of global errno.
- One second later, thread_2 print errno value. However, thread_2 print **its own errno** instead of global errno.

```

invistd@server:~/share$ gcc ./hello.c -o ./hello -pthread
invistd@server:~/share$ ./hello
func_1, errno value: 1
func_2, errno value: 0
invistd@server:~/share$

```

Return value from pthreads functions

In pthreads API, functions return 0 on success or a positive value on failure. The returned failure value is used to replace errno.