Faculty of Computer Science & Engineering

Operating Systems

Nguyen Minh Tri
nmtribk@hcmut.edu.vn
302-B9

Ho Chi Minh City
University of Technology

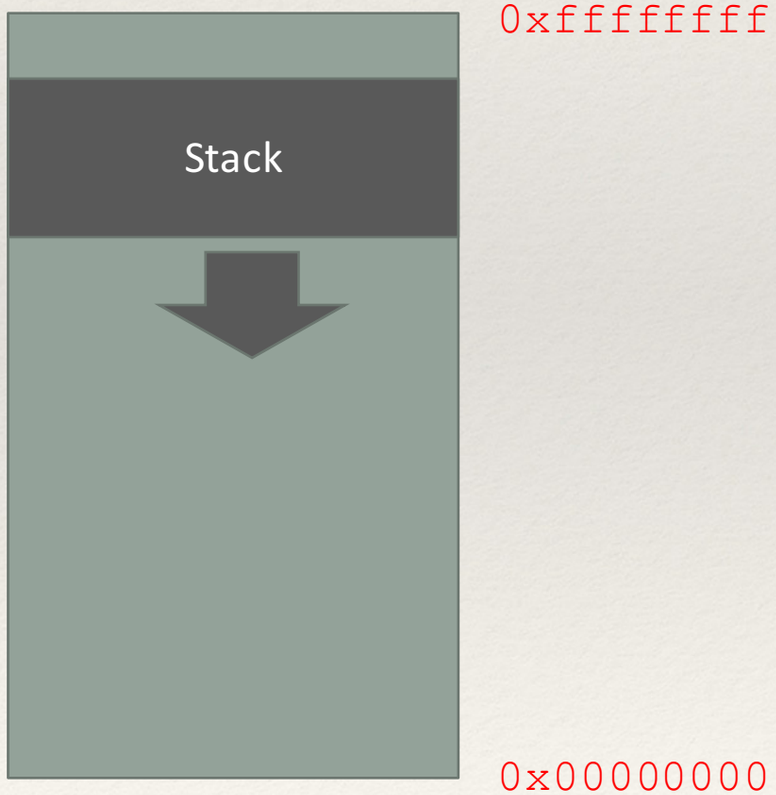
Lab 5 - Thread

Objective

- Distinguish thread and process
- Understand stack organization
- Multi-thread programming

Stack

- Stack is one of hotspot memory region of a program.
- Stack is used to store local data and call information for nested procedure calls.
- Stack grows downward from it origin.



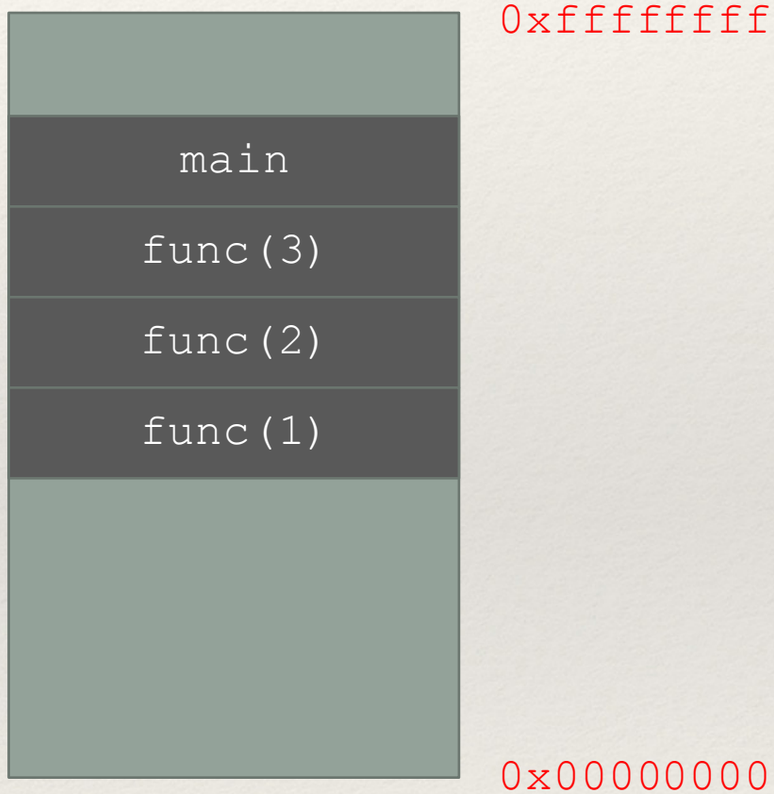
Multiple thread programming

- Exercise: Compile and run the program below.

```
#include <stdio.h>
void func (unsigned long number) {
  unsigned long local_f = number;
  printf("#%2l --> %p\n", local_f, &local_f);
  local_f--;
  if (local_f > 0) {
  func(local_f);
  }
}
int main() {
  func(10);
}
```

Stack

- An example of calling func(3) from main.
- Sample ouput:
  - #3 --> 0x7fffed45352c
  - #2 --> 0x7fffed4534fc
  - #1 --> 0x7fffed4534cc



Stack

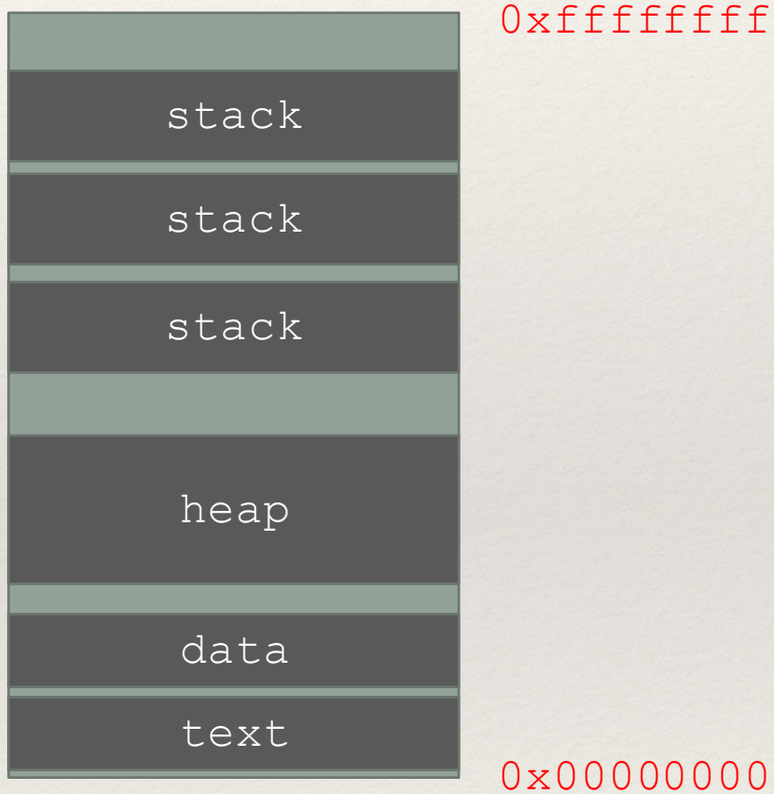
- Similar to heap, stack space is limited.
- Stack size is typically 4 or 8MB.
- To show the default stack size in Linux, typing ulimit -s from command line interface.
- When the size of stack exceed its limit, stack overflow error occurs.

Stack

- Sources of stack overflow errors
  - Infinite recursion
  - Very deep recursion
    - Use small stack
    - Very large size stack (local) variables
- Exercise: Modify the previous program to force stack overflow error to occur (Hint: Be aware printf spam).

Stack

- Unlike other segments, we could have multiple stacks in one process. Each stack belong to one thread.
- Stack is a private region and only visible to the thread currently using it.



Thread in Linux

- Thread is a basic unit of CPU utilization. Thread can be managed independently by scheduler.
- In most cases, a thread is a component of a process.
- Multiple threads can exist within one process, execute concurrently and share resources.
- Each thread has their own program counter, register set, and a stack.

Why thread?

- The benefits of multi threading
- Responsiveness
- Resource sharing
- Economy
- Scalability

Multiple thread programming

- Initially, each process has one thread (i.e. your main function) at the beginning. But we can let it create other threads (remember fork?) by using APIs given by the OS.

- In Linux we create new threads through POSIX pthread functions.In Linux we create new threads through POSIX pthread functions.

End

Thanks!