CSCI315 – Operating Systems Design

Department of Computer Science
Bucknell University

Process Structure

Ch 3

This set of notes is based on notes from the textbook authors, as well as L. Felipe Perrone, Joshua Stough, and other instructors.
Xiannong Meng, Fall 2021.

To fit the lab schedule, we discussed the process creation using fork() first.

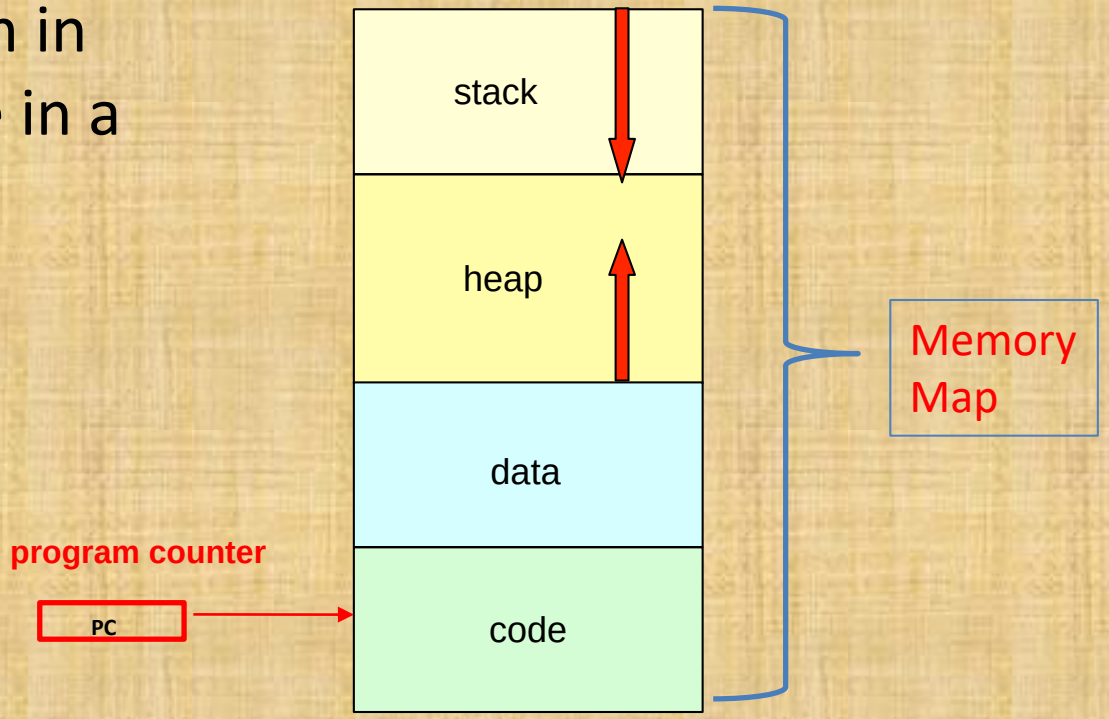
The topics in this set of slides logically should’ve come before the process creation.

What is a process?

- A process is a program in execution.
  - It is a dynamic element, while program (code) is static.
  - A process contains code, data, heap, and stack segments. In particular, a process has a program counter that indicates which line of code is to be executed next.
  - All these information needs to be stored and updated while the process is executing.

Process Concept

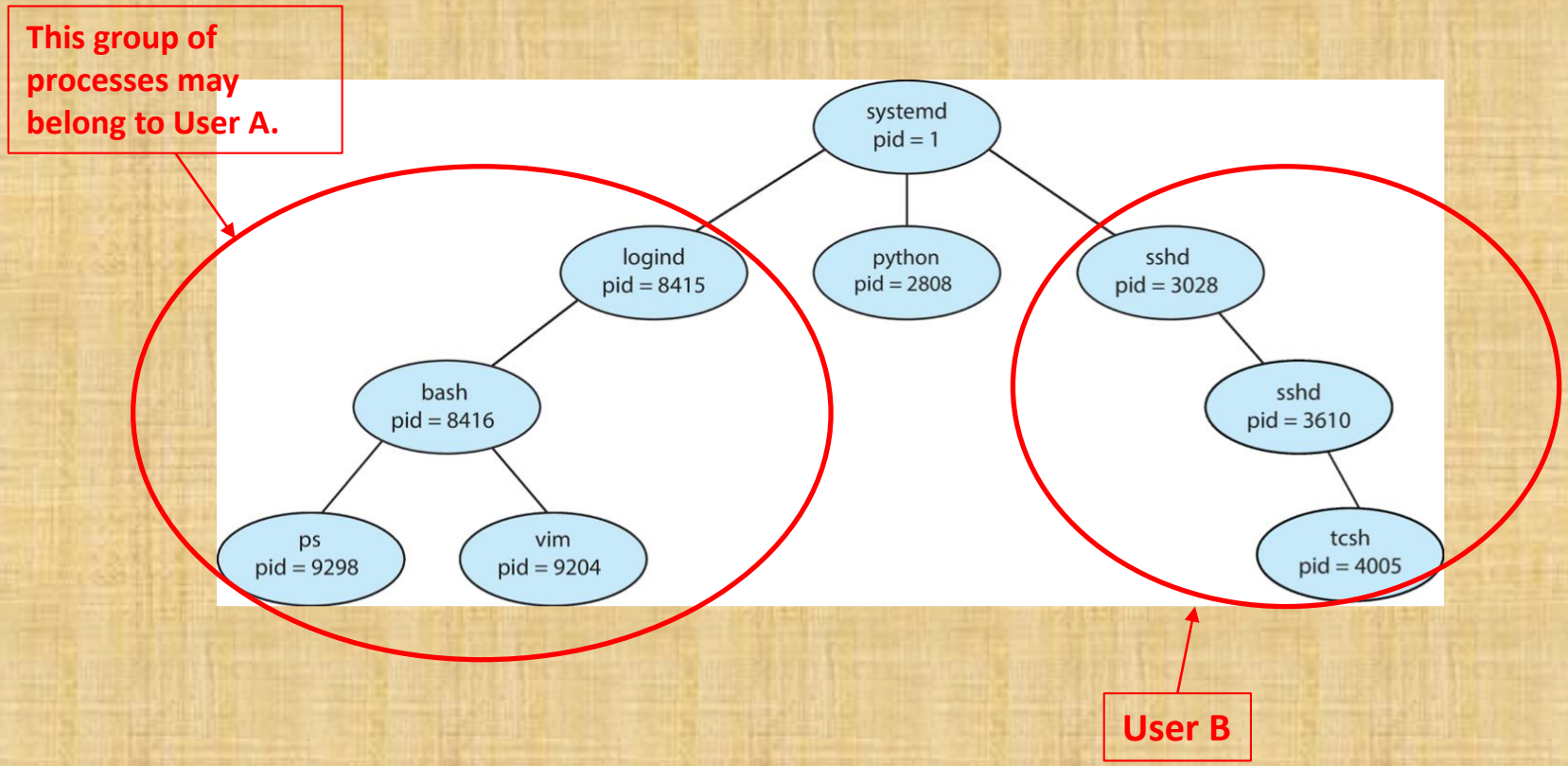
- Process – a program in execution; the code in a process executes sequentially.
- A process includes:
  - program counter,
  - code,
  - stack,
  - heap,
  - data section.



Linux Process Control Commands

- `ps(1)` // process status
- `top(1)` // display processes
- `htop(1)` // process viewer
- `pstree(1)` // display a process tree
- `kill(1)` // terminate a process

Sample Processes in Linux



Find out processes on Linux

Try the command **`ps -aef | less`**

xmeng@linuxremote2:~

File Edit View Search Terminal Help

```
UID          PID   PPID  C STIME TTY          TIME CMD
root           1      0  0 May30 ?        00:00:37 /usr/lib/systemd/sy
root           2      0  0 May30 ?        00:00:00 [kthreadd]
root           4      2  0 May30 ?        00:00:00 [kworker/0:0H]
root           6      2  0 May30 ?        00:00:00 [ksoftirqd/0]
root           7      2  0 May30 ?        00:00:00 [migration/0]
root           8      2  0 May30 ?        00:00:00 [rcu_bh]
root           9      2  0 May30 ?        00:04:55 [rcu_sched]
root          10      2  0 May30 ?        00:00:00 [lru-add-drain]
root          11      2  0 May30 ?        00:00:07 [watchdog/0]
root          12      2  0 May30 ?        00:00:07 [watchdog/1]
root          13      2  0 May30 ?        00:00:00 [migration/1]
root          14      2  0 May30 ?        00:00:00 [ksoftirqd/1]
root          16      2  0 May30 ?        00:00:00 [kworker/1:0H]
root          17      2  0 May30 ?        00:00:07 [watchdog/2]
root          18      2  0 May30 ?        00:00:00 [migration/2]
root          19      2  0 May30 ?        00:00:00 [ksoftirqd/2]
root          21      2  0 May30 ?        00:00:00 [kworker/2:0H]
root          22      2  0 May30 ?        00:00:07 [watchdog/3]
```

User ID

Process ID

Parent process ID

CPU time

Command that started the process

What about process 0?

- The processes in the table on the previous page starts at Process 1, what about Process 0?

The operating system kernel identifies each **process** by its **process** identifier. **Process 0** is a special **process** that is created when the system boots; after forking a child **process** (**process** 1), **process 0** becomes the swapper **process** (sometimes also known as the "idle task").

https://en.wikipedia.org/wiki/Parent_process

Process Control Block (PCB)

A data structure that maintains process information.
On Linux, a PCB is implemented as a C structure.

<u>OS bookkeeping information associated with each process:</u>

- Process ID
- Process state,
- Program counter,
- CPU registers,
- CPU scheduling information,
- Memory-management information,
- Accounting information,
- I/O status information,

⋮

process id
process state
program counter
registers
memory limits
list of open files
⋮

How Does PCB Look Like in C?

- PCB is a very complicated data structure.
- Take a look at this file and search for the key word `task_struct` (line 657-1409)
- https://elixir.bootlin.com/linux/latest/source/include/linux/sched.h
- The PCBs of all current processes form a **doubly linked list** maintained by the OS for their management.

A Portion of a PCB (`task_struct`)

File Edit Options Buffers Tools C Help

Save Undo

```
/* Real parent process: */
struct task_struct __rcu    *real_parent;

/* Recipient of SIGCHLD, wait4() reports: */
struct task_struct __rcu    *parent;

/*
 * Children/sibling form the list of natural children:
 */
struct list_head        children;
struct list_head        sibling;
struct task_struct      *group_leader;

/*
 * 'ptraced' is the list of tasks this task is using ptrace() on.
 *
 * This includes both natural children and PTRACE_ATTACH targets.
 * 'ptrace_entry' is this task's link on the p->parent->ptraced list.
 */
struct list_head        ptraced;
struct list_head        ptrace_entry;

/* PID/PID hash table linkage. */
struct pid          *thread_pid;
struct hlist_node       pid_links[PIDTYPE_MAX];
struct list_head        thread_group;

struct completion       *vfork_done;

/* CLONE_CHILD_SETTID: */
int   user          *set_child_tid;
```

-:--- linux-sched.h 42% (875,0) (C/l Abbrev)

V5.13.13 2021-08-29
Lines 874-905

Source: https://elixir.bootlin.com/linux/latest/source/include/linux/sched.h

Processes and OS Queues