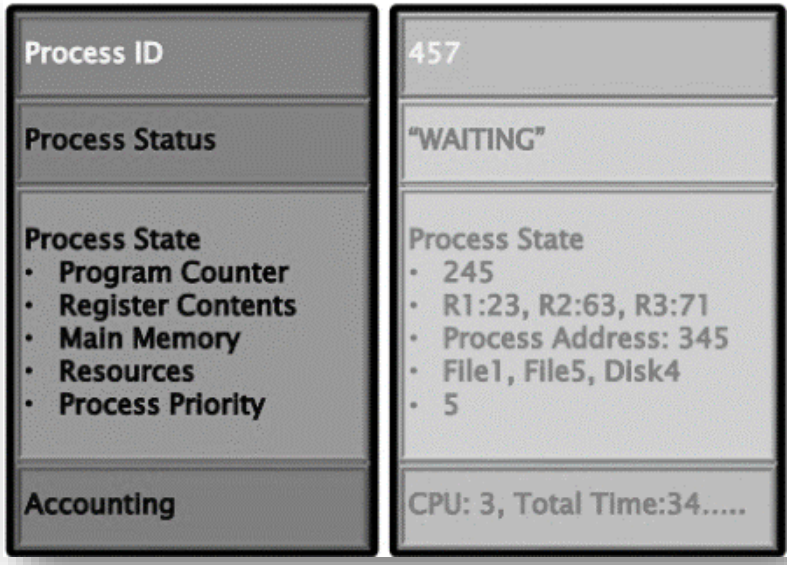
Process -2

CS3600 Spring2022

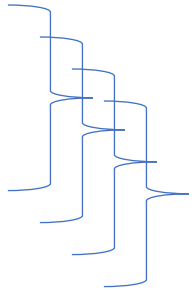
Review

- When an exe file runs – OS creates a process.
- OS timeshares CPU across processes (Virtualize CPU)
- OS has a CPU scheduler
  - Policy: which process will run
  - Mechanism: How to context switch.



Operations on Processes

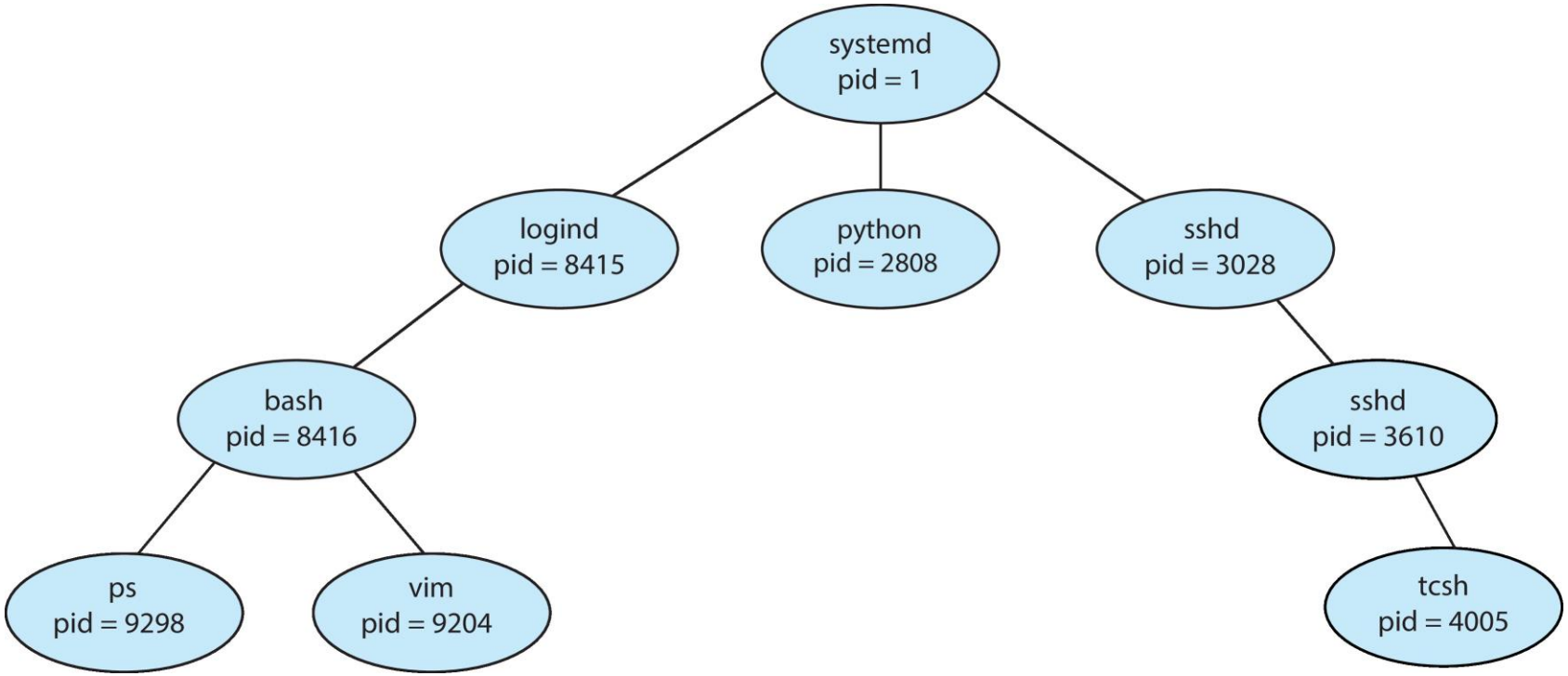
- Process Creation
  - Parent Process
  - Child Process



Tree of processes

Process ID (pid)

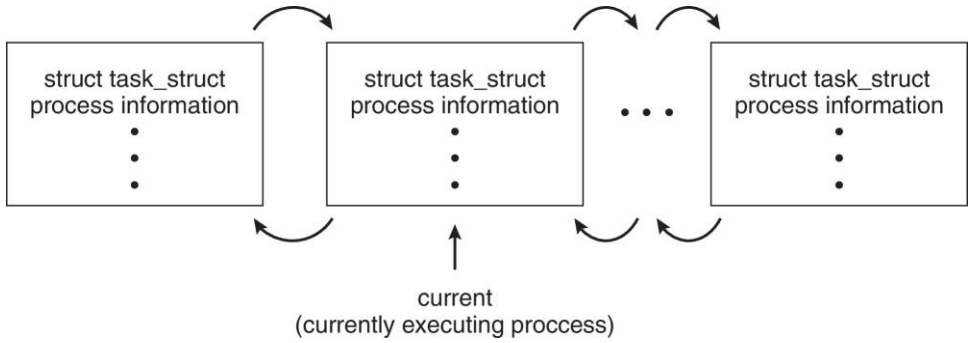
A Tree of Processes in Linux



Process Representation in Linux

Represented by the C structure `task_struct`

```
pid_t pid;                      /* process identifier */
long state;                     /* state of the process */
unsigned int time_slice         /* scheduling information */
struct task_struct *parent;     /* this process's parent */
struct list_head children;      /* this process's children */
struct files_struct *files;     /* list of open files */
struct mm_struct *mm;           /* address space of this process */
```

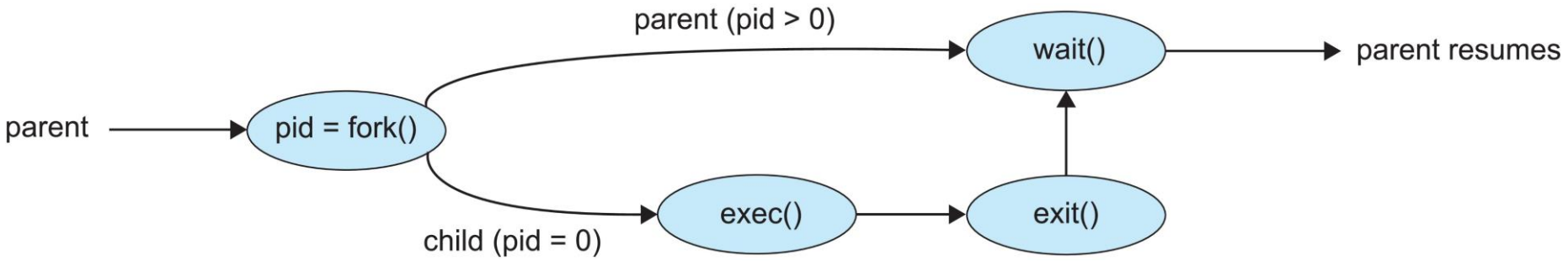


Process Creation

- New Process- possibilities for execution
  - The parent runs concurrently with its children
  - The parent waits until some or all its children have terminated.
- New Process- possibilities for address space
  - Child process has the same program and data section.
  - Child program has a new program loaded in it.

Process Creation

- UNIX examples
  - **`fork()`** system call creates new process
  - **`exec()`** system call used after a **`fork()`** to replace the process' memory space with a new program
  - Parent process calls **`wait()`** for the child to terminate



Process Termination

- Process executes last statement and then asks the operating system to delete it using the **`exit()`** system call.
  - Returns status data from child to parent (via **`wait()`**)
  - Process' resources are deallocated by operating system

- Parent may terminate the execution of children processes using the **`abort()`** system call. Some reasons for doing so:
  - Child has exceeded allocated resources
  - Task assigned to child is no longer required
  - The parent is exiting and the operating systems does not allow a child to continue if its parent terminates

Process Termination

- Some operating systems do not allow child to exists if its parent has terminated. If a process terminates, then all its children must also be terminated.
  - **cascading termination.** All children, grandchildren, etc. are terminated.
  - The termination is initiated by the operating system.
- The parent process may wait for termination of a child process by using the **`wait()`** system call. The call returns status information and the pid of the terminated process

```
pid = wait(&status);
```

- If no parent waiting (**`wait() not called yet`**) process is a **zombie**
- If parent terminated without invoking **`wait`**, process is an **orphan**

Demo

- Lab 2 – Process Creation

Q2

```
int value=5;
int main()
{
  pid_t pid;

        pid=fork();

        if(pid==0){
         value+=15;
         return 0;
        }

      else if(pid>0) {
        wait(NULL);
        printf("Parent: value =%d ",value);
        return 0;
       }
      }
```

What value will be printed for the variable ‘value’ ?

Lab2 – in class work as team of 2

- Turn in screen shots of outputs in terminal for all programs in a single file in Lab2 classwork

- Write a C program to check if the given number of arguments is correct, if not prompt the user to do the correct format.

- Which command can list all the running processes?

- Write a C program to create a process, print process id for Parent and child.
  - Modify the program to assign a variable in the program , assign a new value in child and print the value both in child and parent process.

Announcements as on (02/03/22)

- Read Module 2.2,2.3
- Lab 2 due on 02/04
- Weekly Quiz2 and first homework will be open on Friday (02/04)