The process Part 1

Objectives:

- Understand the concept of “process”,
- Know the principles of management and process control;
- Learn basic process manipulation commands

What is a process?

- You write a program in Pascal, C or JAVA language…etc,
- You start the compilation,
- At the end, you start the execution of this program:

➢ At this stage, we talk about processes (task, process, task, job).

A process is a running program.

Multi-task systems

- Linux is a multitasking system;

- A multi-task system is a system that ensures the execution of several programs at the same time (several processes);

- In this type of systems, the processor (CPU) is shared between the different processes residing in memory →they are ready to be executed,

Multi-task systems

The basic principle is described as follows:

- the processor (CPU) is allocated to a process residing in memory for a fixed duration.
- After the end of the fixed duration, the system will choose (by election principle) the next process to execute among several waiting processes;
- Waiting processes are queued;

→Hence a process has a momentary life (from its creation until its termination; it passes from one state to the other before its end of execution.

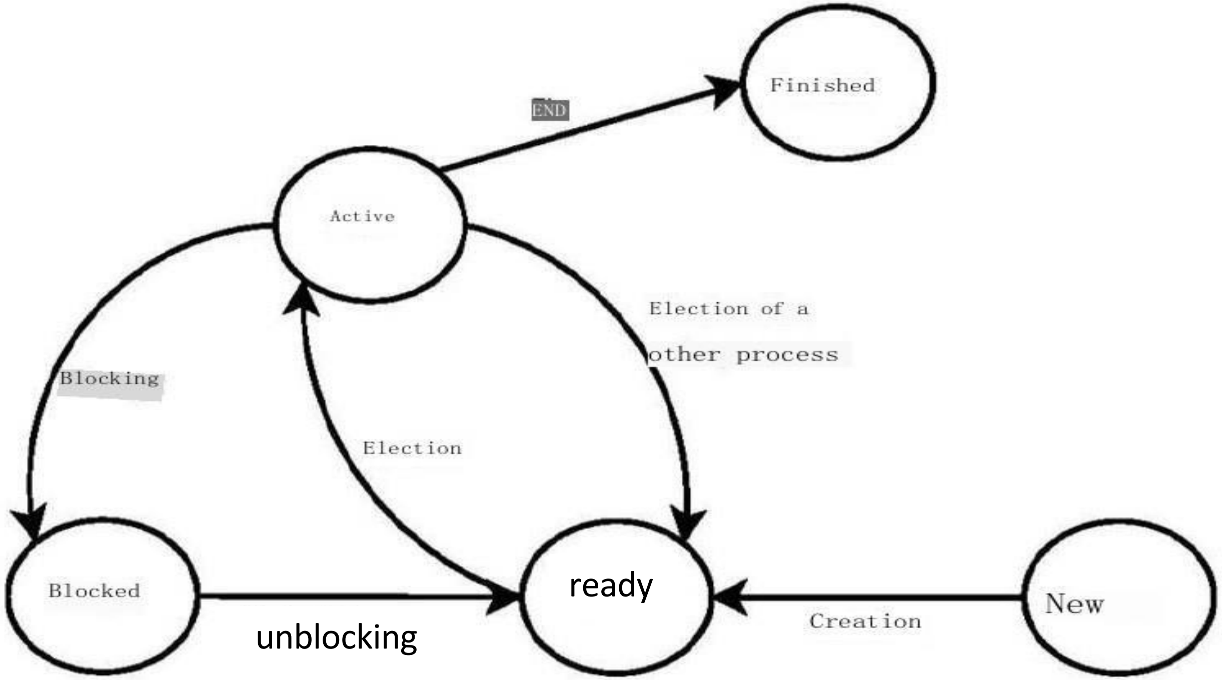
States of a process

- During the life of a process, it can be in different states:
  - Active: the process controls the processor, it is running;
  - Blocked (waiting): the process is suspended, however the central processor (CPU) cannot reactivate it as long as the process is waiting for an external event which can unblock it. For example, a process does an input/output operation (waiting for data from keyboard),
  - Ready (or passive): the process is temporarily suspended, it waits for the processor to be freed (CPU executes the instructions of another process), it is put on hold in RAM.

States of a process

1. A process is created (new).
2. A process is passive Or ready;
3. A process active (the CPU executes the program instructions whose data it has in central memory).

4. A process is blocked(he is waiting for data).
5. A process is destroyed (finished).



The role of an operating system in process management

- It is the OS which must ensure complete management of the creation, destruction and state transitions of a process.

- It is always up to the OS to allocate the memory space necessary for the residence of each process.

- It is still the OS which ensures the organization and inter-process communication.

Processes in Linux

Linux kernel launchesba first process (init) when starting the machine.

➢ this process is the ancestor (the father) of all processes launched on the Linux system;

➢ This process creates child processes which builds a tree structure.

→All processes running on a Linux system are organized in the form of a tree.

Processes in Linux

Each process has:

- A number (its PID,*Process ID*, “process identifier”),
- The identifier of its parent (PPID,*Parent Process ID*, “parent process identifier”) the one who created it.
- Its owner: user number (IUD), and group number (GIUD),
- The processing time (TIME), the reference to a working directory.
- The particularity of a process is to run with rights granted to the user who initiated the command; (important for system security).

The process tree in Linux

❑ The first process init (having a PID=1) launches the following process:

➢The process called “login», he is responsible for opening the session.

- The shell process (bash) is the child of the login process (the process that is waiting for a command to interpret it):

✓ In the case of an internal command, it will not launch a new process (since it is internal to the shell).

✓ In the case of an external command, the shell process launches a *new process*(execution of the program linked to the command launched).

Remark:

Built-in (internal) commands are integrated directly into the shell and are executed more quickly (cd, echo,pwd), while external commands are separate programs that require the creation of a separate process for their execution (ls,grep,sed). Both types of commands are essential in the use of Linux, and each has its own advantages and specific use cases.

Example

The command **pstree** displays the current process tree of execution:

```
$pstree
init ├
       |- - -
       ├login---bash+
                     |pstree (it's an external command)
```

Process control:

The ps command

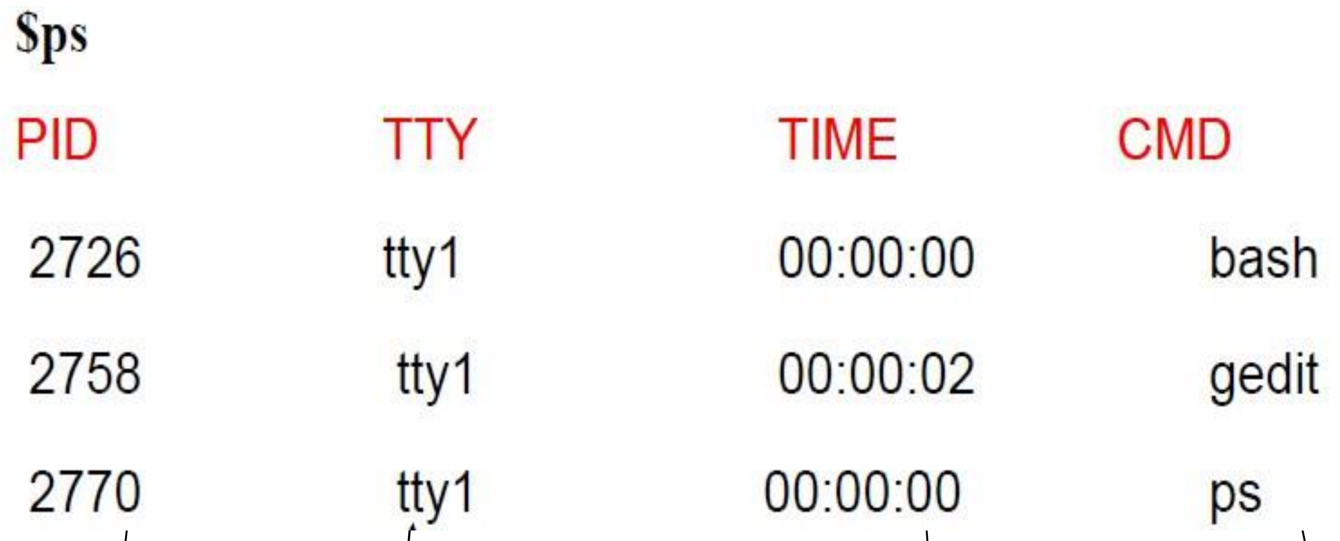
The “ps” (Process status) command allows you to display information about the process ;

Syntax: ps [options]

Some options:

- -a : Shows processes associated with a terminal,
  - -A: Shows all processes on the system, including those without a terminal.
  - x: show processes that do not have a controlling terminalt ttyn: list of processes on the ttyn terminal
  - u: show the processes running under the specified user,
  - f: option adds additional columns to the output, including the parent process ID (PPID), the controlling terminal (TTY)

Example 1



```
$ps
PID      TTY      TIME        CMD
 2726    tty1     00:00:00      bash
 2758     tty1    00:00:02      gedit
 2770     tty1    00:00:00      ps
```

identifier of process

Associated Terminal

CPU consumed time

The command

Displays only **the processes related to the current terminal** (TTY), which typically includes the shell and any commands run within it.

Example Output:

Let's imagine a system with the following processes:

- `bash` (a terminal session)
- `nginx` (a system daemon running in the background)
- `cron` (another background system process)

Running the following commands:

1. **`ps -a`**: This might show processes like:

```
bash

PID   TTY       TIME   CMD

1234  pts/0     00:00  bash

5678  pts/0     00:00  ps
```

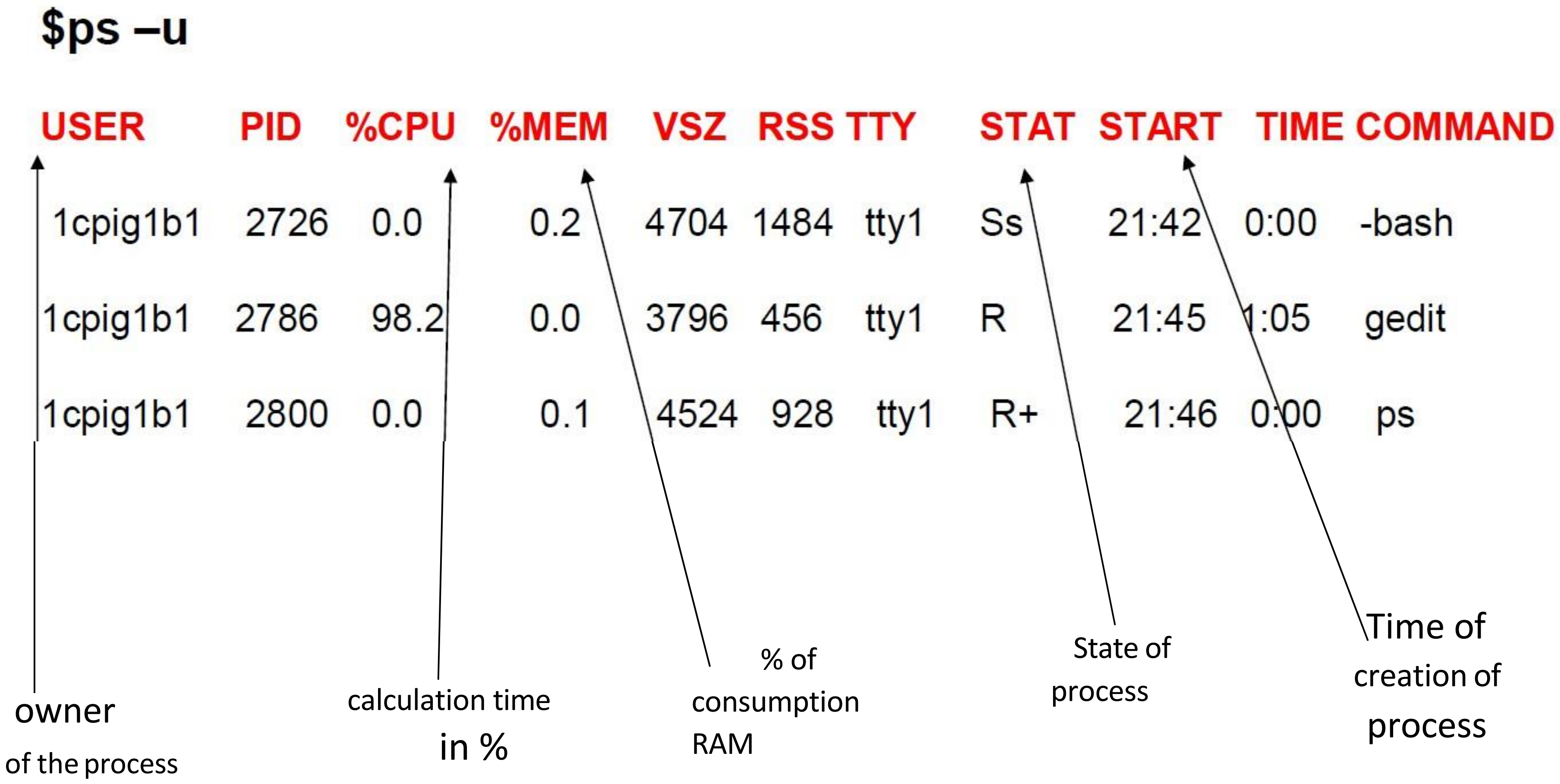
`2.ps -A`: This would show all processes of all terminals and those without terminals:

```
bash

PID   TTY       TIME   CMD

1234  pts/0     00:00  bash

4321  ?         00:00  nginx

5678  pts/0     00:00  ps

9101  ?         00:00  cron
```

As you can see, `ps -A` lists processes like `nginx` and `cron`, which aren't associated with a terminal, while `ps -a` excludes them.

Example 2

- This option allows you to show processes belonging to a specific user.
- You can provide the **username** or the **user ID (UID)**.



```
$ps –u

USER        PID   %CPU  %MEM   VSZ   RSS TTY     STAT  START   TIME COMMAND

 1cpig1b1   2726   0.0      0.2   4704  1484  tty1   Ss     21:42  0:00   -bash
1cpig1b1   2786   98.2      0.0   3796   456   tty1   R      21:45  1:05   gedit
1cpig1b1    2800   0.0      0.1   4524   928   tty1   R+     21:46  0:00   ps
```

owner of the process

calculation time in %

% of consumption RAM

State of process

Time of creation of process

Example 3

Default Behavior: If run without additional options, ps -f shows only processes associated with your current terminal session.

Excludes background processes: System daemons and processes without a terminal (unless combined with other options like -x).

```
ps -Af # Full details of all processes on the system

ps -f -u username # Full details of processes for a specific user
```