“Київський фаховий коледж зв’язку”

Циклова комісія Комп’ютерної інженерії

**ЗВІТ ПО ВИКОНАННЮ**

**ЛАБОРАТОРНОЇ РОБОТИ №4**

з дисципліни: «Операційні системи»

**Тема: «** **Команди Linux для**

**Управління процесами»**

Виконала студентка

групи РПЗ-13б

Дімітрова С.П.

Перевірив викладач

Сушанова В.С.

Київ 2024

**Мета роботи:**

1. Отримання практичних навиків роботи з командною оболонкою Bash.
2. Знайомство з базовими командами для управління процесами.

**Матеріальне забезпечення занять**

1. ЕОМ типу IBM PC.
2. ОС сімейства Windows та віртуальна машина Virtual Box (Oracle).
3. ОС GNU/Linux (будь-який дистрибутив).
4. Сайт мережевої академії Cisco netacad.com та його онлайн курси по Linux

**Короткі теоретичні відомості:**

**Monitoring Programs**

Peeking at the processes

When a program runs on the system, it's referred to as a process. To examine these processes, you need to become familiar with the ps command, the Swiss Army knife of utilities. It can produce lots of information about all the programs running on your system.

Unfortunately, with this robustness comes complexity — in the form of numerous parameters — making the ps command probably one of the most difficult commands to master. Most system administrators find a subset of these parameters that provide the information they want, and they stick with using only those.

That said, however, the basic ps command doesn't really provide all that much information:

$ ps

PID TTY TIME CMD

3081 pts/0 00:00:00 bash 3209 pts/0 00:00:00 ps

$

Not too exciting. By default, the ps command shows only the processes that belong to the current user and that are running on the current terminal. In this case, we had only our bash shell running (remember, the shell is just another program running on the system) and, of course, the ps command itself.

The basic output shows the process ID (PID) of the programs, the terminal (TTY) that they are running from, and the CPU time the process has used.

The GNU ps command that's used in Linux systems supports three different types of com¬mand line parameters:

■ Unix-style parameters, which are preceded by a dash

■ BSD-style parameters, which are not preceded by a dash

■ GNU long parameters, which are preceded by a double dash

The work.following sections examine the three different parameter types and show examples of how they.

*The ps Command Unix Parameters*

|  |  |
| --- | --- |
| Parameter | Description |
| -A | Shows all processes |
| -N | Shows the opposite of the specified parameters |
| -a | Shows all processes except session headers and processes without a terminal |
| -d | Shows all processes except session headers |
| -e | Shows all processes |
| -C cmslist | Shows processes contained in the list *cmdlist* |
| -G grplist | Shows processes with a group ID listed in *grplist* |
| -U userlist | Shows processes owned by a userid listed in *userlist* |
| -g grplist | Shows processes by session or by groupid contained in *grplist* |
| -p pidlist | Shows processes with PIDs in the list *pidlist* |
| -s sesslist | Shows processes with session ID in the list *sesslist* |
| -t ttylist | Shows processes with terminal ID in the list *ttylist* |
| -u userlist | Shows processes by effective userid in the list *userlist* |
| -F | Uses extra full output |
| *-*O format | Displays specific columns in the list *format,* along with the default columns |
| -M | Displays security information about the process |
| -c | Shows additional scheduler information about the process |
| -f | Displays a full format listing |
|  | Shows job information |
| -l | Displays a long listing |
| *-o* format | Displays only specific columns listed in *format* |
| -y | Prevents display of process flags |
| -Z | Displays the security context information |
| -H | Displays processes in a hierarchical format (showing parent processes) |
| -n namelist | Defines the values to display in the WCHAN column |
| -w | Uses wide output format, for unlimited width displays |
| -L | Shows process threads |
| -V | Displays the version of ps |

That's a lot of parameters, and there are still more! The key to using the ps command is not to memorize all the available parameters — only those you find most useful. Most Linux system administrators have their own sets of parameters that they use for extracting pertinent information. For example, if you need to see everything running on the system, use the -ef parameter combination (the ps command lets you combine parameters like this):

**$ ps –ef**

Quite a few lines have been cut from the output to save space, but you can see that lots of processes are running on a Linux system. This example uses two parameters: the -e parameter, which shows all the processes running on the system, and the -f parameter, which expands the output to show a few useful columns of information:

■ UID: The user responsible for launching the process

■ PID: The process ID of the process

■ PPID: The PID of the parent process (if a process is started by another process)

■ C: Processor utilization over the lifetime of the process

■ STIME: The system time when the process started

■ TTY: The terminal device from which the process was launched

■ TIME: The cumulative CPU time required to run the process

■ CMD: The name of the program that was started

This produces a reasonable amount of information, which is what many system administrators want to see. For even more information, you can use the -l parameter, which produces the long format output:

**$ ps -l**

Notice the extra columns that appear when you use the -l parameter:

■ F: System flags assigned to the process by the kernel

■ S: The state of the process (O = running on processor; S = sleeping; R = runnable, waiting to run; Z = zombie, process terminated but parent not available; T = process stopped)

■ PRI: The priority of the process (higher numbers mean lower priority)

■ NI: The nice value, which is used for determining priorities

■ ADDR: The memory address of the process

■ SZ: Approximate amount of swap space required if the process was swapped out

■ WCHAN: Address of the kernel function where the process is sleeping

Real-time process monitoring

The ps command is great for gleaning information about processes running on the system, but it has one drawback. The ps command can display information only for a specific point in time. If you're trying to find trends about processes that are frequently swapped in and out of memory, it's hard to do that with the ps command.

Instead, the top command can solve this problem. The top command displays process information similarly to the ps command, but it does it in real-time mode. Figure 1 is a snapshot of the top command in action.

The first section of the output shows general system information. The first line shows the current time, how long the system has been up, the number of users logged in, and the load average on the system.

The load average appears as three numbers: the 1-minute, 5-minute, and 15-minute load averages. The higher the values, the more load the system is experiencing. It's not uncommon for the 1-minute load value to be high for short bursts of activity. If the 15-minute load value is high, your system may be in trouble.

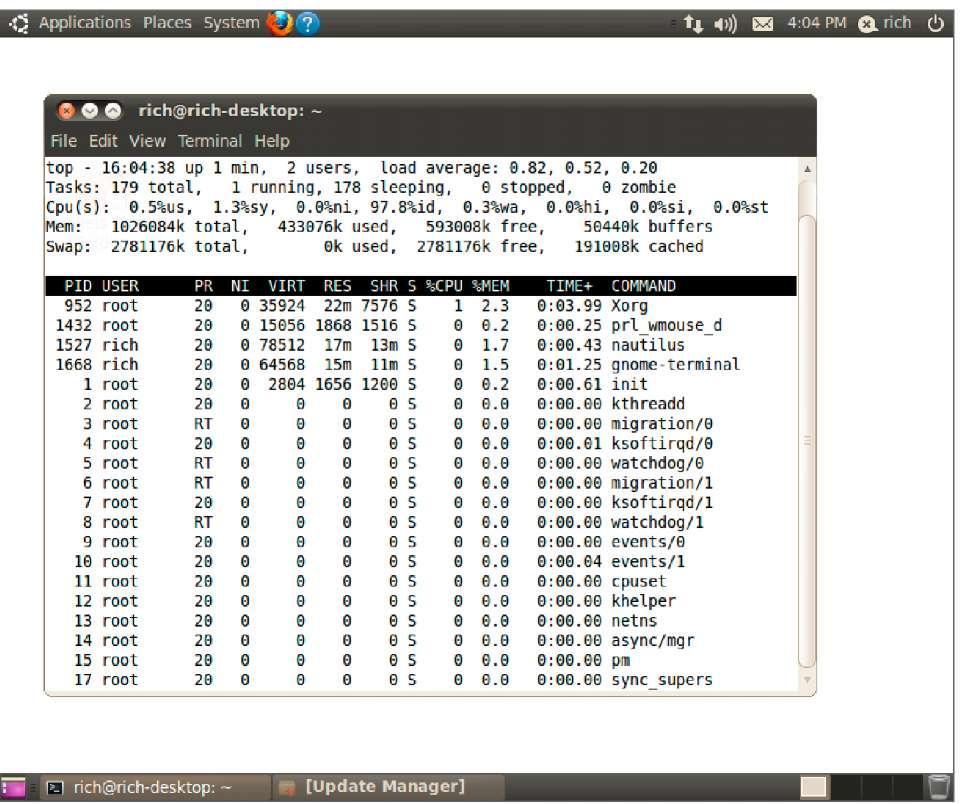


Figure 1. The output of the ***top*** command while it is running

The second line shows general process information (called tasks in top): how many processes are running, sleeping, stopped, and zombie (have finished but their parent process hasn't responded).

The next line shows general CPU information. The top display breaks down the CPU utilization into several categories depending on the owner of the process (user versus system processes) and the state of the processes (running, idle, or waiting).

Following that are two lines that detail the status of the system memory. The first line shows the status of the physical memory in the system, how much total memory there is, how much is currently being used, and how much is free. The second memory line shows the status of the swap memory area in the system (if any is installed), with the same information.

Finally, the next section shows a detailed list of the currently running processes, with some information columns that should look familiar from the ps command output:

■ PID: The process ID of the process

■ USER: The user name of the owner of the process

■ PR: The priority of the process

■ NI: The nice value of the process

■ VIRT: The total amount of virtual memory used by the process

■ RES: The amount of physical memory the process is using

■ SHR: The amount of memory the process is sharing with other processes

■ S: The process status (D = interruptible sleep, R = running, S = sleeping, T = traced or stopped, or Z = zombie)

■ %CPU: The share of CPU time that the process is using

■ %MEM: The share of available physical memory the process is using

■ TIME+: The total CPU time the process has used since starting

■ COMMAND: The command line name of the process (program started)

By default, when you start top, it sorts the processes based on the %CPU value. You can change the sort order by using one of several interactive commands while top is running. Each interactive command is a single character that you can press while top is running and changes the behavior of the program. Pressing f allows you to select the field to use to sort the output, and pressing d allows you to change the polling interval. Press q to exit the top display.

You have lots of control over the output of the top command. Using this tool, you can often find offending processes that have taken over your system. Of course, after you find one, the next job is to stop it, which brings us to the next topic.

Stopping processes

A crucial part of being a system administrator is knowing when and how to stop a process. Sometimes, a process gets hung up and needs a gentle nudge to either get going again or stop. Other times, a process runs away with the CPU and refuses to give it up. In both cases, you need a command that allows you to control a process. Linux follows the Unix method of interprocess communication.

In Linux, processes communicate with each other using signals. A process signal is a predefined message that processes recognize and may choose to ignore or act on. The developers program how a process handles signals. Most well-written applications have the ability to receive and act on the standard Unix process signals.

*Linux Process Signals*

|  |  |  |
| --- | --- | --- |
| Signal | Name | Description |
| 1 | HUP | Hangs up |
| 2 | INT | Interrupts |
| 3 | QUIT | Stops running |
| 9 | KILL | Unconditionally terminates |
| 11 | SEGV | Produces segment violation |
| 15 | TERM | Terminates if possible |
| 17 | STOP | Stops unconditionally, but doesn't terminate |
| 18 | TSTP | Stops or pauses, but continues to run in background |
| 19 | CONT | Resumes execution after STOP or TSTP |

Two commands available in Linux allow you to send process signals to running processes.

The kill command

The kill command allows you to send signals to processes based on their process ID (PID). By default, the kill command sends a TERM signal to all the PIDs listed on the command line. Unfortunately, you can only use the process PID instead of its command name, making the kill command difficult to use sometimes.

To send a process signal, you must either be the owner of the process or be logged in as the root user.

**$ kill 3940**

-bash: kill: (3940) - Operation not permitted $

The TERM signal tells the process to kindly stop running. Unfortunately, if you have a runaway process, most likely it ignores the request. When you need to get forceful, the -s parameter allows you to specify other signals (either using their name or signal number).

As you can see from the following example, no output is associated with the kill command.

**#kill -s HUP 3940**

**#**

To see if the command was effective, you must perform another ps or top command to see if the offending process stopped.

The killall command

The killall command is a powerful way to stop processes by using their names rather than the PID numbers. The killall command allows you to use wildcard characters as well, making it a very useful tool when you have a system that's gone awry:

# killall http\*

#

This example kills all the processes that start with http, such as the httpd services for the Apache web server..

**Завдання для попередньої підготовки.**

1. \*Прочитайте короткі теоретичні відомості до лабораторної роботи та зробіть невеличкий словник базових англійських термінів з питань призначення команд та їх параметрів.

|  |  |
| --- | --- |
| Термін англійською | Термін українською |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

1. На базі розглянутого матеріалу дайте відповіді на наступні питання:
   1. \*Які команди для моніторингу стану процесів ви знаєте. Як переглянути їх можливі параметри?
   2. \*Чи може команда ps у реальному часі відслідковувати стан процесів?
   3. \*\*За якими параметрами можливе сортування процесів в команді top? Як переключатись між ними?
   4. \*\*Які команди для завершення роботи процесів ви знаєте?
2. Прочитати матеріал про роботу з процесами у терміналі:
3. [Процеси в Linux. Управління процесами](https://acode.com.ua/processes-in-linux/)
4. [Find out what processes are running in the background on Linux](https://www.cyberciti.biz/faq/find-out-what-processes-are-running-in-the-background-on-linux/)

**Хід роботи.**

* 1. Початкова робота в CLI-режимі в Linux ОС сімейства Linux:
  2. Запустіть віртуальну машину VirtualBox, оберіть CentOS та запустіть її. Виконайте вхід в систему під користувачем: CentOS, пароль для входу: reverse ***(якщо виконуєте ЛР у 401 ауд.)*** та запустіть термінал.
  3. Запустіть віртуальну машину Ubuntu\_PC ***(якщо виконуєте завдання ЛР через академію netacad)***
  4. Запустіть свою операційну систему сімейства Linux ***(якщо працюєте на власному ПК та її встановили)*** та запустіть термінал.
  5. Дайте відповіді на наступні питання:
* Як вивести вміст директорії /proc? Де вона знаходиться та для чого призначена? Охарактеризуйте інформацію про її вміст?
* Як вивести інформацію про поточні сеанси користувачів. Якою командою це можна зробити?
* Які дії можна зробити в терміналі за допомогою комбінацій Ctrl + C, Ctrl + D та Ctrl + Z?
* \*Чим відрізняється фоновий процес від звичайного. Де вони використовуються?
* \*Опишіть наступні команди та поясніть що вони виконують – команда jobs, bg, fg.
* \*\*Якою командою можна переглянути інформацію про запущені в системи фонові процеси та задачі?
* \*\*Як призупинити фоновий процес, як його потім відновити та при необхідності перезапусти?
  1. Запустіть термінал, та в командному рядку виконайте наступні дії для ознайомлення з роботою з процесами:
* запустіть команду top, проаналізуйте отриманий в цій команді результат та охарактеризуйте найбільш активні процеси у системі;
* призупинити виконання команди top (треба використати комбінацію клавіш);
* вивести інформацію про процеси за допомогою команди ps;
* \*наведіть 5 прикладів з використанням різних параметрів команди ps (наприклад, вивести тільки системні процеси, вивести процеси конкретного користувача, вивести дерево процесів тощо). Опишіть, що саме роблять обрані Вами параметри
* \*\*передивіться чи є у Вас запущені фонові процеси, які саме?
* \*\*відновити виконання призупиненого фонового процесу спочатку у позиції “на передньому плані” (foreground), потім ще раз його призупинити, а потім відновити його виконання у позиції “на задньому плані” (background)
* завершити роботу даного фонового процесу.

**Відповіді на контрольні запитання**

1. Яке призначення директорії */proc* в системах Linux. Яку інформацію вона зберігає?
2. Як серед будь-яких трьох процесів динамічно визначати, який з них в поточний момент часу використовує найбільший обсяг пам'яті? Який відсоток пам’яті він споживає від загального обсягу?
3. Як отримати ієрархію батьківських процесів в системах Linux? Наведіть її структуру та охарактеризуйте.
4. \*Чим відрізняється команда *top* від *ps*?
5. \*Які додаткові можливості реалізує *htop* в порівнянні з *top*?
6. \*\*Опишіть компоненти вашої мобільної ОС для здійснення моніторингу запущених в системі процесів?
7. \*\*Чи підтримує Ваша мобільна ОС термінальне керування роботою процесів, опишіть як саме.
8. \*\*Чи можливо поставити сторонні програмні засоби, що дозволяють організувати управління та моніторинг роботою процесів у Вашому мобільному телефоні. Коротко опишіть їх.

**Висновки:**