

Міністерство освіти і науки України
Львівський національний університет імені Івана Франка
Факультет електроніки

Звіт
про виконання лабораторної роботи №3
на тему: «Процеси в ОС Unix та оволодіння ними.
Оволодіння практичними навичками професійної
роботи з командною оболонкою shell – використання
змінних і створення командних файлів.»

Виконав студент
групи ФЕІ - 23
факультету електроніки
Попович Василь
Викладач:
Сінькевич О. О.

Частина 1:

Мета: Оволодіння практичними навичками роботи з процесами — створення і знищення, керування процесами і їхній аналіз

Теоретична частина: UNIX – багатозадачна система з розділенням часу. Це означає, що в системі одночасно виконується багато процесів. Кожний процес асоційований з певним користувачем, від імені якого цей процес діє. Для того, щоби переглянути список процесів, існує команда `ps`. Ця команда має багато ключів-модифікаторів, які визначають, яку саме інформацію про процеси повинна виводити команда. Слід зазначити, що в різних системах UNIX значення цих ключів може суттєво відрізнитись. Типові ключі: `-a` виводить інформацію про всі процеси, а не лише про процеси даного користувача, `-l` та `-x` виводять розширену інформацію про процес. Кожний процес в системі має свій унікальний ідентифікатор – PID. За цим ідентифікатором можна звертатись до процесу. Крім того, кожний процес виникає не сам по собі – він має так званий батьківський процес, що характеризується ідентифікатором PPID (parent process ID). Таким чином утворюється ієрархія процесів, що бере початок від початкового процесу `init`. Створення нового процесу у системі UNIX виконується у два етапи. Спочатку системний виклик `fork()` викликає “розщеплення” поточного процесу на два тотожні (різниця буде лише у тому, що процес-“нащадок” має інші PID і PPID). Далі виконується системний виклик `exec()`, який “підміняє” контекст поточного процесу іншим контекстом. Розглянемо типовий приклад, коли один процес має запустити на виконання інший – командна оболонка `sh` виконує команду `ls`, для чого утворює новий процес, в якому виконується програма `/usr/bin/ls`.

Завдання до виконання:

1. Переглянути процеси для відповідного користувача можна за допомогою команди `ps` з параметром `u`.

```
vova@Lenovo:~$ ps u
```

USER	PID	%CPU	%MEM	VSZ	RSS	TTY	STAT	START	TIME	COMMAND
vova	10440	0.3	0.1	25440	4232	pts/0	Ss	18:42	0:00	/bin/bash
vova	10953	0.0	0.0	20608	1248	pts/0	R+	18:43	0:00	ps u

2. За допомогою команди **ps** з параметрами **auxf** можна вивести процеси для усіх користувачів. Команда **more** дозволяє посторінкове про листування.

```
vova@Lenovo:~$ ps auxf | more
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         2  0.0  0.0      0     0 ?        S    10:15   0:00 [kthreadd]
root         3  0.0  0.0      0     0 ?        S    10:15   0:00 \_ [ksoftirqd/0]
root         4  0.0  0.0      0     0 ?        S    10:15   0:00 \_ [kworker/0:0]
root         6  0.0  0.0      0     0 ?        S    10:15   0:00 \_ [migration/0]
root         7  0.0  0.0      0     0 ?        S    10:15   0:00 \_ [watchdog/0]
root         8  0.0  0.0      0     0 ?        S    10:15   0:00 \_ [migration/1]
root        10  0.0  0.0      0     0 ?        S    10:15   0:00 \_ [ksoftirqd/1]
root        11  0.0  0.0      0     0 ?        S    10:15   0:00 \_ [kworker/0:1]
root        12  0.0  0.0      0     0 ?        S    10:15   0:00 \_ [watchdog/1]
root        13  0.0  0.0      0     0 ?        S    10:15   0:00 \_ [migration/2]
root        15  0.0  0.0      0     0 ?        S    10:15   0:00 \_ [ksoftirqd/2]
root        16  0.0  0.0      0     0 ?        S    10:15   0:00 \_ [watchdog/2]
root        17  0.0  0.0      0     0 ?        S    10:15   0:00 \_ [migration/3]
root        18  0.0  0.0      0     0 ?        S    10:15   0:00 \_ [kworker/3:0]
root        19  0.0  0.0      0     0 ?        S    10:15   0:00 \_ [ksoftirqd/3]
root        20  0.0  0.0      0     0 ?        S    10:15   0:00 \_ [watchdog/3]
root        21  0.0  0.0      0     0 ?        S<   10:15   0:00 \_ [cpuset]
root        22  0.0  0.0      0     0 ?        S<   10:15   0:00 \_ [khelper]
root        23  0.0  0.0      0     0 ?        S    10:15   0:00 \_ [kdevtmpfs]
```

3. У списку процесів відображається процес `/bin/bash`, який відповідає за запусканий термінал. Командою **kill -9 <PID>** ми можемо закрити даний процес.

```
vova      6501  1.3  0.5 499556 21144 ?        Sl   10:20   0:00 pantheon-terminal
vova      6504  0.0  0.0  14788   828 ?        S    10:20   0:00 \_ gnome-pty-helpe
vova      6505  0.5  0.1  25440  4228 pts/0    Ss   10:20   0:00 \_ /bin/bash
vova      7150  0.0  0.0  20720  1240 pts/0    R+   10:20   0:00 \_ ps auxf
vova@Lenovo:~$ kill -9 6505
```

4. Переглянути список задач можна командою **jobs**. Я попередньо ввів команду **sleep 120 &**, для того щоб продемонструвати роботу команди **jobs**.

```
vova@Lenovo:~$ sleep 120 &
[1] 12818
vova@Lenovo:~$ jobs
[1]+  Running                  sleep 120 &
vova@Lenovo:~$
```

- 5-6. Замість команди вказаної у методичці, я використав команду **sleep 100 &**

(тому що вона виконується дуже малий проміжок часу).

Опис нижче приведенного рисунку:

Перша команда запускає процес. Далі виводиться список процесів за допомогою команди **jobs**. Командою **fg** даний процес переводиться у

пріоритетний. Призупиняється процес комбінацією клавіш **Ctrl+Z**. Знову виводимо список процесів і бачимо, що процес зупинено. Командою **bg** процес переводиться у фоновий режим, таким чином, відновлюючи свою роботу.

```
vova@Lenovo:~$ sleep 100 &
[1] 22027
vova@Lenovo:~$ jobs
[1]+  Running                  sleep 100 &
vova@Lenovo:~$ fg
sleep 100
^Z
[1]+  Stopped                  sleep 100
vova@Lenovo:~$ jobs
[1]+  Stopped                  sleep 100
vova@Lenovo:~$ bg
[1]+  sleep 100 &
vova@Lenovo:~$ jobs
[1]+  Running                  sleep 100 &
vova@Lenovo:~$
```

7. Командою **pgrep -v -u <user>** виводимо список процесів, які запущені не від імені цього користувача.

```
vova@Lenovo:~$ pgrep -v -u root
800
802
827
829
949
971
973
1285
```

8. Вивести календар за 2002 рік через 1 хвилину після моменту введення можна за допомогою команди **sleep 60 ; cal 2002**

```
vova@Lenovo:~$ sleep 20 ; cal 2002
                2002
    Январь      Февраль      Март
Вс Пн Вт Ср Чт Пт Су  Вс Пн Вт Ср Чт Пт Су  Вс Пн Вт Ср Чт Пт Су
      1  2  3  4  5          1  2          1  2
  6  7  8  9 10 11 12    3  4  5  6  7  8  9    3  4  5  6  7  8  9
13 14 15 16 17 18 19    10 11 12 13 14 15 16    10 11 12 13 14 15 16
20 21 22 23 24 25 26    17 18 19 20 21 22 23    17 18 19 20 21 22 23
27 28 29 30 31          24 25 26 27 28          24 25 26 27 28 29 30
                                     31

    Апрель      Май          Июнь
Вс Пн Вт Ср Чт Пт Су  Вс Пн Вт Ср Чт Пт Су  Вс Пн Вт Ср Чт Пт Су
      1  2  3  4  5  6          1  2  3  4          1
  7  8  9 10 11 12 13    5  6  7  8  9 10 11    2  3  4  5  6  7  8
14 15 16 17 18 19 20    12 13 14 15 16 17 18    9 10 11 12 13 14 15
21 22 23 24 25 26 27    19 20 21 22 23 24 25    16 17 18 19 20 21 22
28 29 30          26 27 28 29 30 31          23 24 25 26 27 28 29
                                     30

    Июль        Август      Сентябрь
Вс Пн Вт Ср Чт Пт Су  Вс Пн Вт Ср Чт Пт Су  Вс Пн Вт Ср Чт Пт Су
      1  2  3  4  5  6          1  2  3          1  2  3  4  5  6  7
  7  8  9 10 11 12 13    4  5  6  7  8  9 10    8  9 10 11 12 13 14
14 15 16 17 18 19 20    11 12 13 14 15 16 17    15 16 17 18 19 20 21
21 22 23 24 25 26 27    18 19 20 21 22 23 24    22 23 24 25 26 27 28
28 29 30 31          25 26 27 28 29 30 31    29 30

    Октябрь     Ноябрь      Декабрь
Вс Пн Вт Ср Чт Пт Су  Вс Пн Вт Ср Чт Пт Су  Вс Пн Вт Ср Чт Пт Су
      1  2  3  4  5          1  2          1  2  3  4  5  6  7
  6  7  8  9 10 11 12    3  4  5  6  7  8  9    8  9 10 11 12 13 14
13 14 15 16 17 18 19    10 11 12 13 14 15 16    15 16 17 18 19 20 21
20 21 22 23 24 25 26    17 18 19 20 21 22 23    22 23 24 25 26 27 28
27 28 29 30 31          24 25 26 27 28 29 30    29 30 31

vova@Lenovo:~$
```

9. Для організування періодичного видалення в домашньому каталозі усіх файлів з розширенням ***.bak** і ***.tmp** потрібно редагувати файл **crontab** для відповідного користувача, найзручніше це робити користуючись командою **crontab**. Для редагування **crontab** для відповідного користувача використовують **crontab -u <user> -e**, щоб переглянути вміст файлу використовують **crontab**

<user> -І, що й зображено на рисунку.

```
vova@Lenovo:~$ crontab -u vova -e
no crontab for vova - using an empty one

Select an editor. To change later, run 'select-editor'.
 1. /bin/nano      <---- easiest
 2. /usr/bin/vim.tiny

Choose 1-2 [1]: 1
crontab: installing new crontab
vova@Lenovo:~$ crontab -u vova -l
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow  command
00 12 * * * rm /home/vova/*.tmp /home/vova/*.bak
vova@Lenovo:~$
```

Висновок: Операційна система Linux надає широкі можливості у управління процесів. Користувач може запускати, зупиняти, призупиняти процеси, отримати вичерпуючу інформацію про будь-який процес.

Частина 2.

Мета: Оволодіння практичними навичками професійної роботи з командною оболонкою shell – використання змінних і створення командних файлів.

Теоретична частина: У попередніх роботах ми вже познайомились з командними оболонками (shell). У цій роботі розглянемо прийоми професійної роботи з командними оболонками, а саме використання змінних оточення і створення командних файлів. Змінні оточення Усі змінні вашого оточення виводяться за допомогою команди set, ознайомтесь з ними. Командні файли Командний файл, або сценарій (також дуже часто кажуть “скрипт” від англійського script – сценарій) є текстовим файлом, який оформлено з дотриманням певних правил, і який містить команди, у найпростішому випадку повністю аналогічні тим командам, що вводяться з клавіатури. Командна оболонка здатна запускати такий файл на виконання і послідовно виконувати команди, що містяться в ньому. Для користувача, що запустив сценарій, його виконання буде виглядати як виконання звичайної програми. Зверніть увагу на розбіжності у різних програмних оболонках shell, які суттєві для програмування. Під час виконання роботи впевніться, в якій із програмних оболонок Ви працюєте (зазвичай, для FreeBSD це csh чи tcsh, а для Linux – bash, який є розвитком sh), і яка буде запускатись для виконання Вашого командного файлу (це визначається першим рядком Вашого командного файлу). Уважно прочитайте правила використання операторів if і формування перевірки відповідної умови. Зверніть увагу на команду test. Важливою можливістю командних оболонок (усіх) є обробка так званих пакетних файлів.

Завдання до виконання:

Написати скрипт для виконання наступного завдання:

Частина 1:

1. Визначити, хто є користувачем системи та виведіть на екран.
2. За допомогою змінних оточення визначити домашній каталог користувача
3. Знайти всі файли, які належать вам у вашому домашньому каталозі
4. За архівувати ці файли з іменем back_up_
5. Якщо архів з даним іменем вже існує, то вивести запит на його перезапис.
6. Встановити права на отриманий архів тільки для читання.

Частина 2:

1. Розархівувати файли з архіву зі збереженням структури каталогів
2. Якщо файли відрізняються у порівнянні з оригіналом, то потрібно виводитися запит на його перезапис.

Частина 1. Командою `touch` було створено файл `task1.tar`, у який я записав наступний сценарій.

```
1  #!/bin/bash
2
3  whoami
4  echo $HOME
5  echo
6  user=$(whoami)
7  echo "Find files with user owner"
8  find $HOME -maxdepth 1 -user $user -type f | cut -d "/" -f 4 | tee list | cat
9
10 echo "Savinf files to arvhive"
11 archive=/home/vova/task1_archive.tar
12 if [ -a "$archive" ]; then
13     echo "Archive exist"
14     echo "Rewrite? "
15     read answer
16     chmod +w "$archive"
17     if [ "$answer = y" ]
18     then
19         tar -cPf "$archive" -T list
20         echo "Files saved in archive"
21     else
22         echo "Dinied"
23     fi
24 else
25     tar -cPf "$archive" -T list
26     echo "Files saved in archive"
27 fi
28 rm list
29
30 chmod 444 "$archive"
31 ls -l "$archive"
```

Варто пояснити деякі моменти коду: перша лінійка є обов'язковою для всіх скриптів. Команда **echo** виводить вказаний текст на екран (аналог `cout` у C++). Для виводу імені користувача використано команду **whoami**. Лінійка 8 виводить список файлів, які знаходяться у домашньому каталозі користувача та записує їх у файл `list`. Опцією `-cPf` ми створюємо архів, зчитуючи файли, які треба за архівувати з файлу `list`. Запустити скрипт можна, надавши йому права для зміни (`chmod +x <file>`), командою `./<file>`

Частина 2: Створено виконуваний файл task2.tar

```
1  #!/bin/bash
2
3  archive=$1
4  if [ -z "$1" ]; then
5      archive=/home/vova/task1_archive.tar
6  fi
7  if [[ ! -a "$archive" ]]; then
8      echo "File doesnt exist"
9      exit
10 fi
11 chmod +w "$archive"
12 extractTo=$2
13 if [ -z "$2" ]; then
14     extractTo=$HOME
15 fi
16 extractTo=$extractTo/tmp
17 mkdir -p "$extractTo"
18 tar xPf "$archive" -C "$extractTo"
19
20 for item in $(ls -1 "$extractTo"); do
21     src="$extractTo/$item"
22     dest="$extractTo/../$item"
23     if [[ -a "$dest" ]] && [[ "$src" -ot "$dest" ]]; then
24         echo "File $item never then archived version"
25         echo "Rewrite? y - Yes"
26         read answer
27         if [ "$answer = y" ]; then
28             cp "$src" "$dest"
29         else
30             echo Denied
31         fi
32     else
33         cp "$src" "$dest"
34     fi
35 done
36
37 rm -rf "$extractTo"
```

Завданням цього скрипта є розархівування файлів, які заархівовано в попередньому завданні.

Висновок: по аналогії роботи з командною стрічкою, можна створити виконуваний файл і записати сукупність часто виконуваних команд, і запускати на виконання даний файл.