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.

Львів – 2015

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 відрізнятись. 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 віповідного 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 **auf** можна вивести процеси для усіх користувачів. Команда **more** дозволяє посторінкове про листування.

```
vova@Lenovo:~$ ps aufx | 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 запущений термінал. Командою **kill -9 <PID>** ми можемо закрити даний процес.

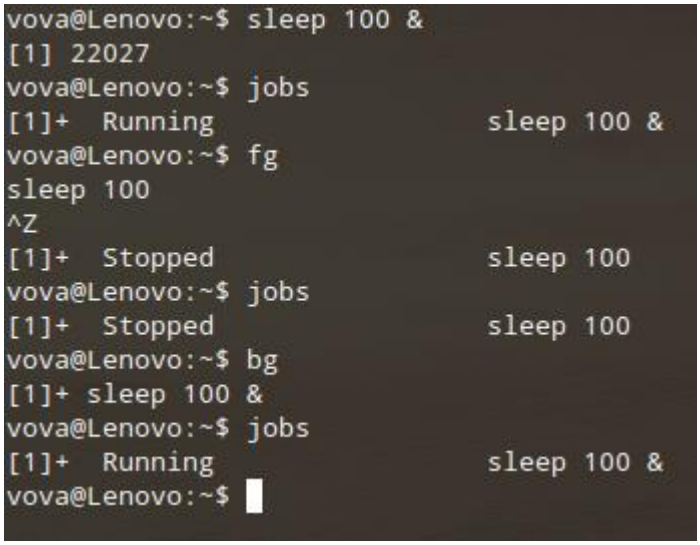
```
vova       6501  1.3  0.5 499556 21144 ?        Sl   10:20   0:00 pantheon-terminal
vova       6504  0.0  0.0  14788   828 ?        S    10:20   0:00  \_ gnome-pty-helpe
vova       6505  0.5  0.1  25440  4228 pts/0    Ss   10:20   0:00  \_ /bin/bash
vova       7150  0.0  0.0  20720  1240 pts/0  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

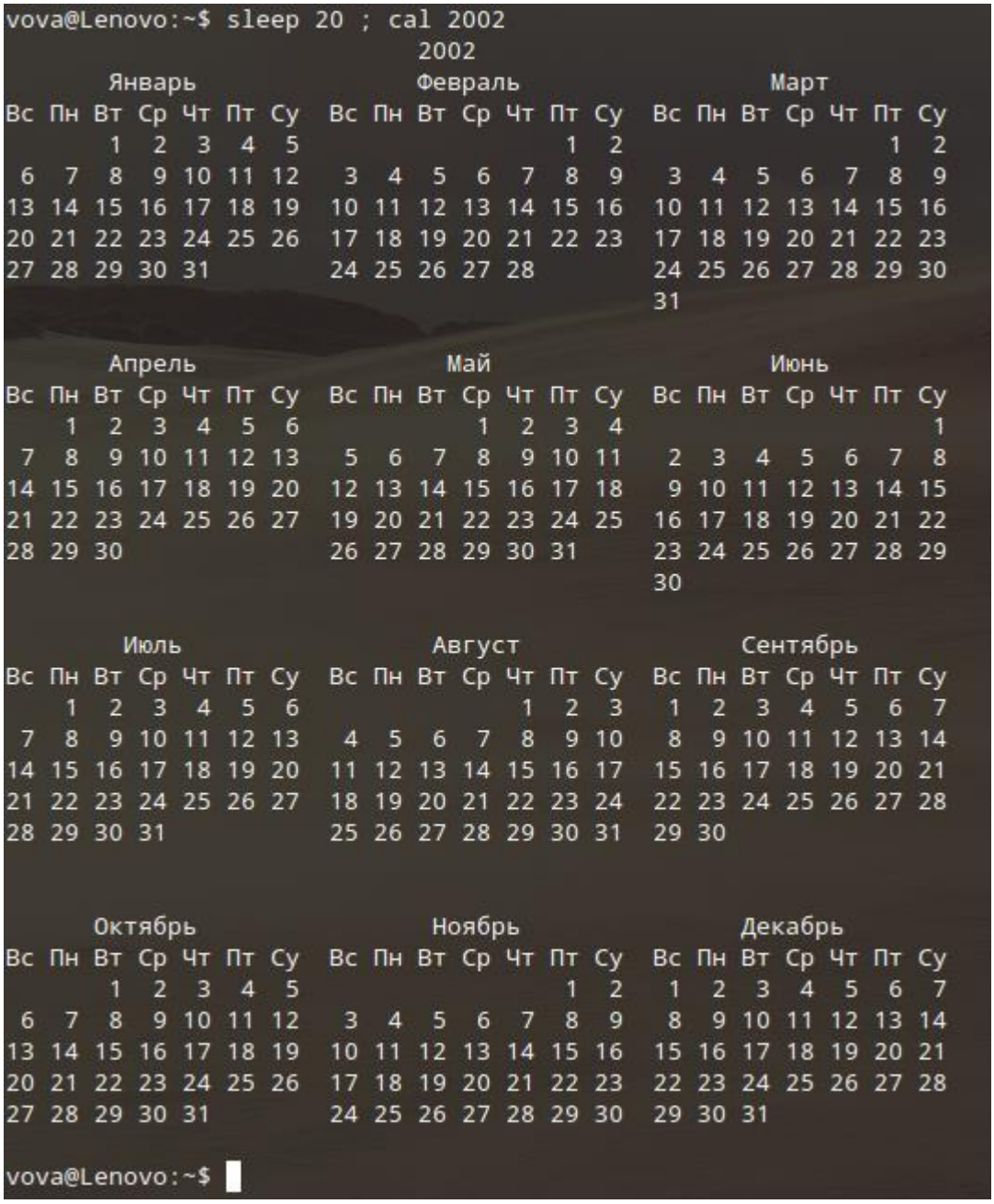
5-6. Замість команди вказаної у методичці, я використав команду **sleep 100 &**
(тому що вона виконується дуже малий проміжок часу).
Опис нижче приведеного рисунку:
Перша команда запускає процес. Далі виводиться список процесів за допомогою команди jobs. Командою **fg** даний процес переводиться у

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



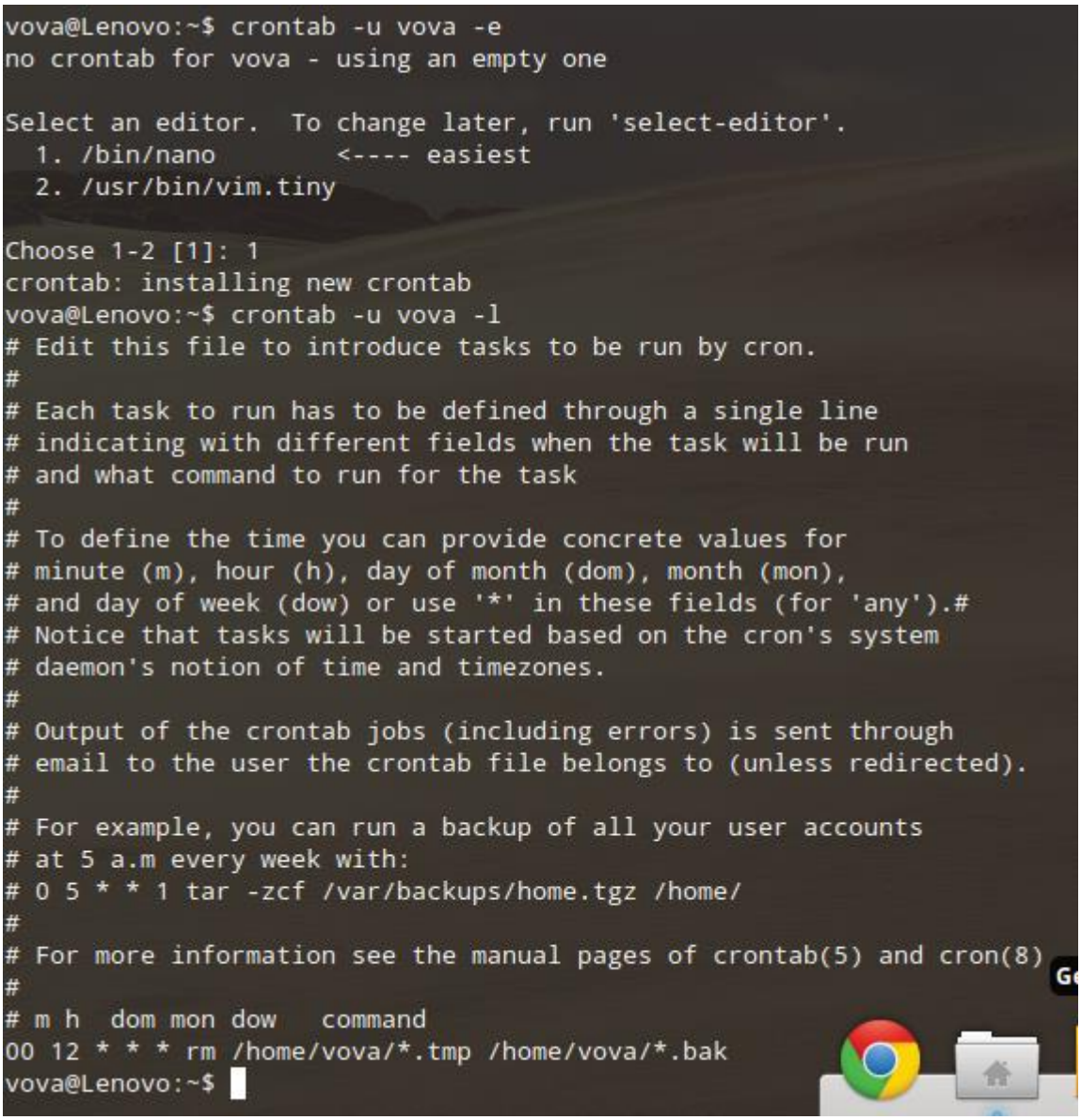
7. Командою **pgrep –v –u <user>** виводимо список процесів, які запущені не від імені цього користувача.

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



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

<user> -l, що й зображено на рисунку.



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 запустивцей 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.

```
#!/bin/bash

whoami
echo $HOME
echo
user=$(whoami)
echo "Find files with user owner"
find $HOME -maxdepth 1 -user $user -type f | cut -d "/" -f 4 | tee list | cat

echo "Savinf files to arvhive"
archive=/home/vova/task1_archive.tar
if [ -a "$archive" ]; then
    echo "Archive exist"
    echo "Rewrite? "
    read answer
    chmod +w "$archive"
    if [ "$answer = y" ]
    then
        tar -cPf "$archive" -T list
        echo "Files saved in archive"
        else
            echo "Dinied"
        fi
    else
        tar -cPf "$archive" -T list
        echo "Files saved in archive"
    fi
    rm list

    chmod 444 "$archive"
    ls -l "$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

```
#!/bin/bash

archive=$1
if [ -z "$1" ]; then
    archive=/home/vova/task1_archive.tar
fi
if [[ ! -a "$archive" ]]; then
    echo "File doesnt exist"
    exit
fi
chmod +w "$archive"
extractTo=$2
if [ -z "$2" ]; then
    extractTo=$HOME
fi
extractTo=$extractTo/tmp
mkdir -p "$extractTo"
tar xPf "$archive" -C "$extractTo"

for item in $(ls -1 "$extractTo"); do
    src="extractTo/$item"
    dest="$extractTo/../$item"
    if [[ -a "$dest" ]] && [[ "$src" -ot "$dest" ]]; then
        echo "File $item never then archived version"
        echo "Rewrite? y - Yes"
        read answer
        if [ "$answer = y" ]; then
            cp "$src" "$dest"
        else
            echo Denied
        fi
    else
        cp "$src" "$dest"
    fi
done

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.