**Комп’ютерний практикум 1. Основи роботи із оболонкою BASH**

Зміст

1. Теоретичні відомості.

2.Завдання

1. **Теоретичні відомості**

Оболонка a6o термінал — је програма, яка забезпечус текстове спілкування між операційною системою та користувачем. Зазвичай је програма в текстовому режимі, яка зчитус введені користувачем дані та інтерпретус ïx як команди системі.

Основні оболонки Linux:

* + нова оболонка Борна (Bash)
  + оболонка С (csh a6o tcsh, покращена csh)
  + оболонка Корна (ksh)
  + Z оболонка (zsh)

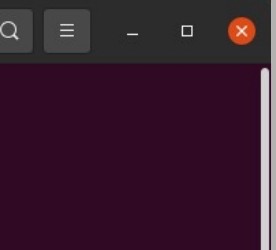
У Linux найпоширенішою с оболонка Bash.

Користувацька оболонка bash може працювати в двох режимах - інтерактивному та неінтерактивному. Відкрити оболонку в Ubuntu можна комбінаціею клавіш Ctrl + Alt + F1. Якщо оболонка видас запрошення (рис. 1), то ви працюете в інтерактивному режимі:

При використанні інтерактивної оболонки користувач вводить команди у командному рядку (prompt). Структура командного рядку наступна:

*username@hostname.’currenf\_directory shell type*

В Ubuntu a6o Debian GNU/Linux рядок для введення команд у звичайного користувача виглядас, яка зображено на рис. 1.



helgl@helgI•pa -

Рис. 1.

Командний рядок складаеться із:

helgi — username — ім’я користувача, який загіустив оболонку.

helgi-pc — hostname — ім’я хоста, на якому працюе оболонка. Існуе також команда ви можете відобразити a6o встановити ім’я хоста системи.

## " " — current directory — директорія, у яшй зараз знаходиться оболонка — ознаиае, що оболонка знаходиться в домаиіньому каталозі поточного користуваиа.

$ — shell\_type — вказуе, що оболонка запускаеться звииайним користувачем. # вказуе, що оболонка запускаеться суперкористувачем root .

## Більшість команд у командному рядку мають однакову базову структуру:

***команда*** *[onцin(’i )/napaмemp(u )...1 fпргумент(и )...]*

## Наприклад:



Рис. 2.

## Команда — програма, яку буде запускати користувач — 1s у наведеному вище прикладі.

**Опція/Параметр** — “перемикаи”, який певним чином змінюе поведінку команди, наприклад -ahl у наведеному вище прикладі.

## Доступ до параметрів можна отримати як у короткому, так i в довгому вигляді. Нагіриклад, -l ідентииний --format=1ong .

Також можна комбінувати кілька варіантів, а для короткой форми літери зазвичай можна вводити разом. Наприклад, yci наступні команди роблять те саме:

### *is -al* is -а -I

*is --all* ***—(ormat——long***

## Аргументи — додаткові дані, які потрібні програмі, як-от ім’я файлу a6o шлях. Наприклад:

*1s -hal /*

## де “/” - аргумент у наведеному вище прикладі, що е посиланням на кореневу директорію.

Сдиною обов’язковою иастиною ціеї структури е сама команда. Загалом, yci інші елементи с необов’язковими, але програма може вимагати вказати певні опції, параметри пи аргументи.

## Більшість команд показують короткий огляд доступних команд, коли вони запускаються з параметром --help . Незабаром ми дізнаемося додаткові способи дізнатися більше про команди Linux.

Типи команд:

* **внутрішні** — команди які е частиною самої оболонки i не е окремими програмами. Таких команд близько 30. Ïx головна мета - виконувати завдання всередині оболонки (наприклад, cd , set , export );
* **зовнішні** — команди які містяться в окремих файлах. Зазвичай ці файли с бінарними програмами a6o скриптами. Коли виконуеться команда, яка не е внутрішньою командою оболонки, оболонка використовуе змінну PATH для пошуку виконуваного файлу з таким же ім’ям, що й команда. На додаток до програм, які встановлюються за допомогою менеджера пакунків дистрибутива, користувачі також можуть створювати власні зовнішні команди.

Команда type показус, якого типу с конкретна команда:

*fype echo*

*echo is а shell builtin fype man*

*man is /usr/biЛman*

Команда echo — је дуже проста i в той же час часто використовувана вбудована команда оболонки Bash. Вона мае тільки одне призначення — виводити рядок тексту в термінал, але застосовусться дуже часто в різних скриптах, програмах, i навіть для редагування конфігураційних файлів.

Команда echo — је не системна утиліта, у неї немае виконуваного файлу. Вона існуе тільки всередині інтерпретатора Bash. Синтаксис команди echo linux:

*echo onцi’i’рядок*

Опції:

* -П — HE ВИВОДИТИ НОВИЙ }ЗЯДОК,
* —е — включити підтримку виведення escape-послідовностей;
* —Е — відключити інтерпретацію escape—послідовностей.

Якщо включена опція —е, можна використовувати наступні escape—послідовності для вставки спеціальних символів:

* + \с — видалити новий рядок;
  + \t — горизонтальна табуляція;
  + \v — вертикальна табуляція;
  + \Ь — видалити попередній символ;
  + \п — новий рядок;
  + \r — символ повернення каретки на початок рядка. Приклади роботи echo

1. Виведеня рядка на екран:

echo Linux Open Source Softwafe Technologies

1. Виведеня змінної:

- **OГOЛOШeHHЯ ЗM1HHOÏ:**

*exporf VAR —hello*

- ВиВедеННя ЗНачеННя зМІННОї

*echo $VAR*

1. ВидалеННя всіх пробілІВ 3 я,Qка (ВикОриСТаННя escape—послідОвНостей):

*echo -е “Linux \bopen \bsource \bsoyware \btechnologies“*

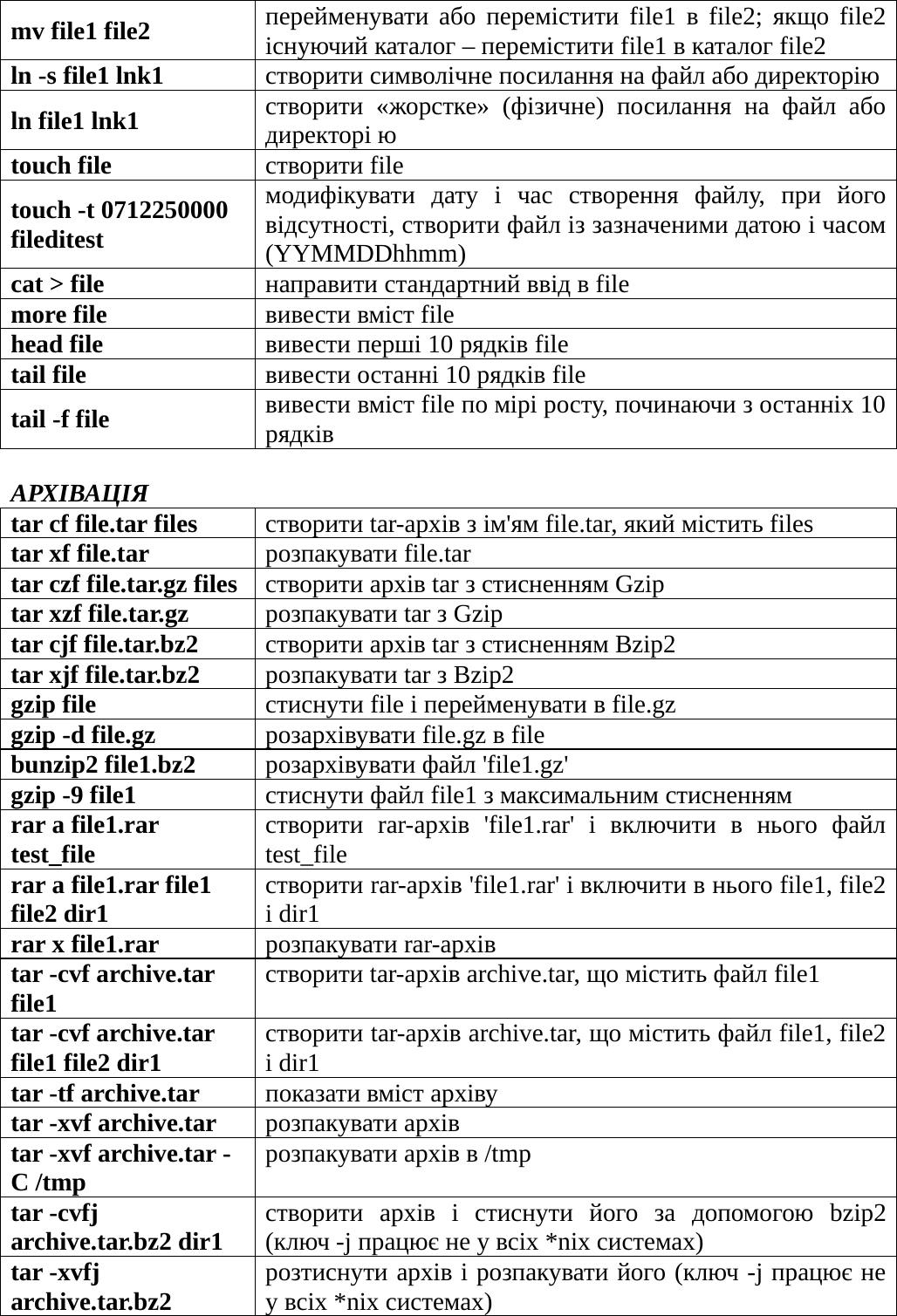
Яхщо рядоК МlСТИть будЬ-які спеціаЛЬНІ СИМВОли a6o Може буТи ЗроЗуМілИМ

іНТерпретаТОроМ НеодНОЗНаЧНО, CЛ1,П BЗflTИ ЙOFO В ЛіlПКИ.

Нижче НаведеНо перелік КОМаНД, яКЇ чаСТО ВИКО|ЗиСТоВуються:

*ФАЙЛОВІ КОМАН,Д,М*

|  |  |
| --- | --- |
| cd ../.. | перейТи В директорію дBaMfi }З1ВНRМИ вище |
| cd | перейТи В ,0,ОМаШНю дирекТорію |
| cd wser | перейТи В ,0,ОМаШНю дирекТорію корисТуВача user |
| gd | перейТи В дирекТорію, В ЯКЇіі ЗНаХОдИЛИСь до переходу В  пОТочНу дирекТорію |
| pwd | покаЗаТи гіоТочНу директорію |
| mkdir dir | СТВО|ЗИТИ КаТаЛОг dif |
| mkdir dirl | СТВО|ЗИТи дирекТорію 3 lM’ЯМ dirl |
| mkdir dirl dir2 | СТВО|ЗИТИ ДВі дирекТорї ОДНОчаСНО |
| mkdir —р  /tmp/dirl/dir2 | СТВО|ЗИТи дереВо директорій |
| rm file | ВН,Я,аЛИТИ file |
| rm —г dir | ВН,Я,іlЛИТИ КіЗТіЗЛОГ dif' |
| rm —f file | ВН,Я,аЛИТИ форсоваНо file |
| rm —rf dir | ВН,Я,аЛИТИ фО|ЗСОВіlНО КаТізЛОг dir |
| rm —f file1 | ВН,Я,аЛИТИ iliiЛ 3 IM’HM 'filel' |
| rmdir dirl | ВН,Я,аЛИТи дирекТорию 3 IM’flM 'dir1' |
| rm —rf dirl | ВН,Я,аЛИТи дирекТорию з 1M’flM 'dirl' i рекурсиВНо Весь ïï  BMICT |
| rm —rf dirl dir2 | ВН,Я,аЛИТИ ,П,Bi дирекТорї i рекурсиВНо Becb ïX BMICT |
| cp file1 file2 | скопіюваТИ filel в file2 |
| cp —г dir1 dir2 | скопіюваТи dirl в dif 2; СТВО]ЭИТЬ KdTilЛOF dir2, якЩО BiH Не  icHyE |
|  | КОПіюВати BCi файли дирекТоріі dir В гіоточНу дирекТорію |
| cp —а /tmp/dirl | КОПіЮВаТи директорію difl 3 yciM BMICTOM В пОТочНу  дирекТорію |
| cp —а dir1 dir2 | КОПіЮВаТи дирекТорію dirl в дирекТорію dif2 |
| mv dirl new\_dir | перейМеНуВаТи a6o переМісТиТи файЛ a6O директорію |



|  |  |
| --- | --- |
| tar -cvfz  **archive.tar.gE dirl** | створити архів i стиснути його за допомогою gzip |
| tar -xvfZ  **archive.tar.gE** | розтиснути архів i розпакувати його |
| **zip filel.zip filel** | створити стиснений zip—apxiв |
| **zip —г filel.Zl]3 file**1  **file2 dirl** | створити стиснений zip—apxiв з включенням в нього  декількох файлів i / a6o директорій |
| **ППE1{ï file1.Zip** | розтиснути i розпакувати zip—apxiв |

Сценарій Bash е простій текстовій файл, якій містить ряд команд для Циклів i запитів. Вони являють собою суміш команд, які вводяться у командному рядку (нагіриклад, ls a6o cp).

Команди оболонки для створення сценарію:

*mkdir 7scripfs cd /scripts* fouch *scripf1.sh*

Далі відкриваемо файл у текстовому редакторі та вводимо команди.

В неінтерактивному режимі оболонка читас команди з деякого файлу i послідовно виконуе ïx. Коли інтерпретатор дійде до кінця файлу, робота з оболонкою автоматично завершиться. Запустити оболонку в неінтерактивному режимі можна з допомогою команд:

*sh script.sh bash script.sh*

scripf.sfi *—* је шлях до файлу, що містить команди для виконання. Такий файл с звичайним текстовим документом, який можна створити з допомогою будь- якого текстового документу. Втім, можна спростити виклик скрипту всього лише зробивши його виконуваним. Для цього необхідно надати відповідні права доступу цьому файлу з допомогою команди chmod. Команда chmod — змінюе права доступу до файлу. Перевірити права доступу до файлу можна командою:

*1s -1* scripf.sfi

Якщо результат не містить літери л на початку, наприклад, —rwxrwxfWX, значить потрібно надати скрипту право бути виконуваним. Для цього потрібно виконати команду:

*chmod и+х script.sh*

У першому рядку скрипта необхідно вказати яка оболонка повинна виконувати цей сценарій. Це можна зробити, розмістивши на початку відповідну вказівку #!/bin/bash. Після цього файл можна буде викликати на

виконання звернувшись до нього в терміналі. Виконуеться скрипт наступним

ЧИНОМ 1



#### Синтаксис мови bash Цитування

Оболонки забезпечують функЦію, яка називаеться цитуванням, вона інкапсулюс задіяні пробіли, спеціальні символи та змінні за допомогою різних типів лапок (” ", ' '). У Bash с три типи Цитування:

* + - подвіЙні лагіки;
    - одинарні лапки;
    - Escape-символи.

Коментарі

Сценарії можуть містити коментарі. Коментарі - је оператори, які можна розміщувати в сценарії оболонки, але який ігноруеться при виконанні. Коментарі повинні починатись з символу # i продовжуються до символу нового рядка. Нагіриклад:

*#!/bin/bash*

*# True script*

Оболонка дозволяе створювати та видаляти змінні, а також виконувати над ними операції. Змінні в bash можуть знаходитись в 3-x областях видимості

1. **Локальні змінні** — це звичайні змінні в всередині одного сценарію. Вони не доступні іншим програмам та сценаріям, які загіускаються з цісї оболонки. Оголошуються змінні з допомогою символу =(зверніть увагу на те, що перед i після = немае пробілів), а до ïx значения звертаються з допомогою символу $:

# *dir —”temp“*

*echo $dir # вивід значения*

# unsef *dir # видалення жінно’і’*

Також можна оголосити *локальну змінну* всередині функції i яка буде доступна лише в тілі Цісї функЦії:

*local var—23*

1. **Змінні оточення** — це змінні, які доступні будь—яким програмам, що запущені з данної оболонки. Оголошуються вони так само як i локальні змінні, але з командою export:

*export* M¥VAR= */Documeлts*

### *echo* $MYVAR *# auaib у термінал зночення змінно’і’*

В bash с багаТо зМіННИХ ОТочеННя, яКЇ Д,ОСИТЬ ЧаСТо зусТрічаюТЬСЯ В

сцеНаріяХ, НаПриКЛаД:

HOME - lliШlX ,0,0 ,D ОМаШНьОго каталогу корисТувача;

PATH - СНИСОК КаТалогіВ, В flKИX ОбО lОНка шукае ВикоНу BaHi файЛи;

PWD - шлях до робочого каталогу;

RANDOM - форМус ціле ВипадкоВе число;

HOSTNAME - IM я коМп ютера, На якоМу ВикоНусТьСя ОбОЛОНКа;

1. Змінні оболонки — це 3МіНН1, яК1 ВСТаНОвлююТЬСя обОлОНКОю i НеобхідНІ Їй для корекТНОЇ ]ЭОбОТИ. Ці ЗМІННі МаюТЬ iMeHa порядкОВОгО HoMepa ($1, $2, $3, ...) I MlCTЯTb аргуМеНТи, Які передаВаЛиСЬ сцеНарію при Запуску, Напрахлад:

./SCfipt.sh datal data2 # ВсередиНі сцеНарію $1='data1', $2='data2'

3MIHHFIM МОЖНа ПрГlСВОіовати ЗНачеННя За ЗаМоВчуВаННяМ:

*$(VAR. —0 J #* Якијо змінна VAR *порожня, npucвo’i’mu ’і’и 0*

ЗМІННі оболОНКИ:

|  |  |
| --- | --- |
| Paramet er | Description |
| $0 | Name of the current shell script |
| $l-$9 | Positional Iarameters l through 9 |
| ${10} | Positional parameter 10 |
| $# | The number of positional parameters |
| $\* | All positional parameters, "$\*" is one string |
| $@ | All positional parameters, "$@" is a set of strings |
| $? | Return status of most recently executed command |
| $$ | Process id of current process |

ПідСТаНОВКа КОмаНд

ПÏ,qCTilHOBKi3 KOMilHД ,D ОЗВОШІС ВИКОјЭиСТоВуваТи ре зулЬТііТ O,ПHICÏ КОМіЗН,ПИ В

КОМаНдНоМу рядку, що МіСТИТЬ іНшу коМаНду. МожНа ВИКО|ЗиСТовуВаТи дВа

альтернативних формату підставновки: $(команда) a6o "команда" (одинарні зворотні лапки). Наприклад,

*#!/bin/bash* lisf- "1s *-1" echo* $lisf

#### Умовні оператори

В скриптовій мові bash підтримуються два оператори розгалуження: if та case. Оператор if, як i в інших мовах, виконус певний блок вказівок, в залежності від умови. Умову огортають в подвійні квадратні дужки [[ ... ]], які bash розглядас як один елемент з кодом виходу. Всередині блоку операторів огорнутого в [[ ]] дозволяеться використовувати оператори && та ||. Приклади:

# *# Однорядковии зanuc*

*i[( ... ]; then echo "true", else echo "[olse”,’ [i,’*

### *## Вкладені умови*

*i[( ... ] && ( ... ]; then*

*eli[ ([ ... &&: ... ]],’* fhen

*else*

Зверніть увагу, що [, умова та ] обов’язково повинні бути розділені пробілами, інакше оболонка сприйме в якості команди [умова.

В нижче наведено можливі умови порівняння: # Робота з файлами

-е Перевірити чи існус файл чи директорія (-f, -d)

-f Файл існуе (!—f - не існуе)

-d Каталог існус (!—f — не існуе)

-s Файл існус i він не порожній

-f Файл існуе i доступний для питания

-w ... на запис

-х ... на виконання

-h С символічним посиланням

# Робота з рядками

-z Порожній рядок

-п Не порожній рядок

—= Рівне

!= Не рівне

# Операції з числами

-eq Рівне

-ne Не рівне

-it Менше

-le Менше a6o рівне

-gt Більше

-ge Більше a6o рівне

Приклади використання умовного оператору:

*i[ ( ”$myvar” -eq 3 ]*

then *echo ”myvar eq 3 as пит”*



then *echo ”myvar eq 3 as srt“*

*i[[ -z $1 ],’ then echo “Empfy Parometer“,’ [i*

Цикли

Мова оболонки дaе користувачеві можливість організовувати циклічне виконання інструкцій з допомогою Циклів:

* while;
* for;
* select.

Оператор while описуеться наступним чином:

*while умова do done*

Інтерпретатор в першу чергу виконус команди, що описані в умові. Якщо результат виконання нульовий, то виконуеться тіло, а, після ïi виконання, перехід до наступної ітерації, в іншому випадку відбуваеться вихід з циклу. В умові може бути будь-яка допустима команда. Приклад:

*#!/bin/sh*

*# Квадрати чисел від 1 до 10*

*while* f *$х —It 10 ] do #значення змінно’і’ х менше 10? echo ${($x\*$x))*

*х— “expr $х + 1 # збіпьшуемо х на 1 done*

Цикл for виконуе тіло для кожного елемента зі списку. Синтаксис циклу for такий:

*[or ім’я in епемент1 елемент2 ...* елементN *do done*

В якості елементів зазвичай використовують різноманітні шаблони (wildcards). Дуже зручно застосовувати for для проходження по каталогах та виконання операцій над групою файлів. В прикладі нижче, цикл проходить по всіх файлах з розширенням \*.bash, переміщуе ïx в директорію /scripts та додае ïx права на виконання.

*#!/bin/sh*

*# Переміщення всіх скриптів з в директорію* /scripts

*[or FILE in $HOMF/\*.bash do mv $FILF $(HOME}7scripfs*

*chmod cx ${HOMF)/scripts7$(FILE) done*

Цикл select допомагас організувати зручне меню вибору i застосовуеться тоді, коли користувач повинен обрати один елемент із запропонованого списку. Загалом Цикл select мае такий самий синтаксис як i цикл fOf:

*select відповідь in елемент1 елемент2 ... елементN do done*

При виконанні цього оператору, всі елементи зі списку висвічуються на екрані зі своїми порядковими номерами у вигляді списку варіантів відповіді, після списку виводиться спеціальне загірошення для введення. Зазвичай воно мае вигляд #?. Введений користувачем номер списку загіисуеться в змінну відповідь. Якщо відповідь містить номер пункту меню, то в змінну заноситься значения відповідного елементу зі списку. Якіцо в списку немас введеного пункту, список буде показаний знову. Після того, як користувач зробить правильний вибір, виконаються вказівки в тілі, а цикл перейде до наступної ітерагці i всі діі повторяться з самого початку — саме тому роботу циклу select бажано переривати.

*#!/bin/sh*

*echo -n ”Введіть назву naкemy. ” && read* PACKAG6

*PS3 — ”Оберіть пакетиий менеджер .’ “ selecf ITEM in bower, npm, pip do*

*case $ITEM in*

*bower) bower install $PACKAGE , , npm) npm* insfO/f *$PACKAG6 „ pip) pip* insfaff *$PACKAGE „*

*esac break*

*done*

Приклад вище запитуе в користувача назву пакету, який він бажас встановити, далі цікавиться який пакетний менеджер використати i в залежності від вибору встановлюе потрібний пакет та зупиняе виконання.

Оболонка також мас команди, для зміни нормального виконання циклу. Оператор bfeak повністю зупиняс виконання циклу, оператор continue - переходить до наступної ітерагці.

#### Завдання

* 1. Написати скрипт який приймас необмежену кількість параметрів та виводить ïx у термінал.
  2. Написати скрипт який зчитуе текстовий файл та виводить кожну його строку окремо. Можливо стане у пригоді команда *cat* та виконання
  3. Написати скрипт який перетворюс файл із формату docx, doc, odt у формат pdf. Ім’я файлу, що конвертуеться, задаеться як параметр при виконанні скрипту. Перевірити чи файл, який конвертусться, існуе. Якііјо ні, то вивести повідомлення. Для конвертування можна використати команду lowriter.
  4. Написати скрипт який перетворюс pdf файл який можна редагувати у pdf файл створений на основі зображень. Для цього потрібно створити тимчасову директорію. Перед Цим потрібно перевірити, чи така директорія не існуе. Якщо так, то видалити ii. Перетворити pdf файл на зображення, які зберегти у тимчасову директорію. Перетворити зображення на pdf файл. Видалити тимчасову директорію. Для перетворення pdf файлу у зображення можна використати команду *pdyoppm.* Для перетворення зображення у pdf можна використати команду convert.