

ПРАКТИЧНА РОБОТА 01

Знайомство з операційною системою Linux та командним рядком bash

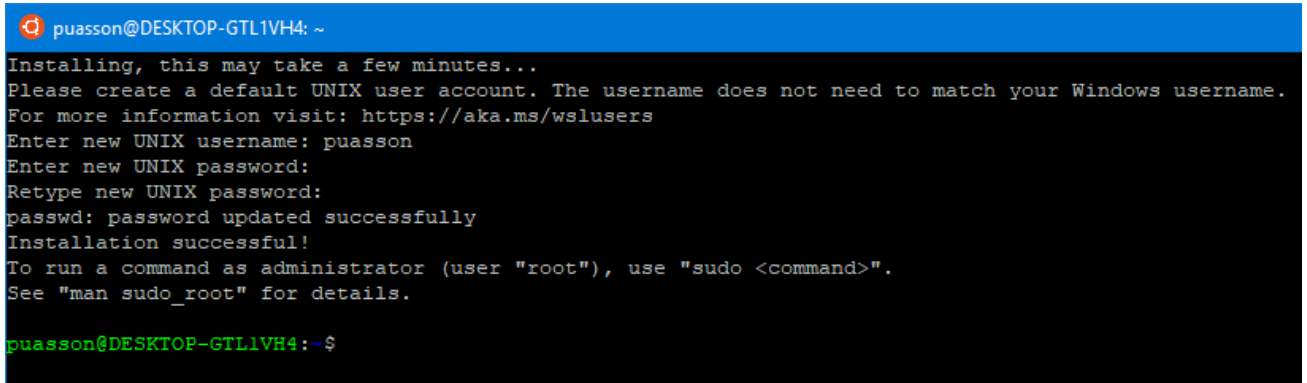
План

1. Операційна система Linux.
2. Командний рядок bash: базові команди.
3. Додаткові можливості командного рядка bash.
4. Розширені можливості командного рядка bash.

1. Операційна система Linux

Для виконання практичної роботи потрібно отримати доступ до командного рядка bash операційної системи Linux. Існує кілька варіантів:

- для Windows 10 – встановити підсистему Linux відповідно до [інструкції](#).



```
puasson@DESKTOP-GTL1VH4: ~
Installing, this may take a few minutes...
Please create a default UNIX user account. The username does not need to match your Windows username.
For more information visit: https://aka.ms/wslusers
Enter new UNIX username: puasson
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
Installation successful!
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.
puasson@DESKTOP-GTL1VH4:~$
```

- Інсталювати віртуальну машину, наприклад VirtualBox, та встановити або розгорнути в ній образ операційної системи Linux. Зокрема, в мережі присутні готові [образи](#) ОС Ubuntu для VirtualBox та VMWare.
- Розгорнути операційну систему на завантажуючому флеш-пристрої, наприклад, як в [інструкції](#).
- Встановити операційну систему Linux разом або замість операційної системи Windows, як пропонує [інструкція](#).

2. Командний рядок bash: базові команди

Сучасні файлові системи мають дерева директорій (папок), у яких директорією є кореневий каталог (root directory, без батьківського каталогу) або субдиректорією (subdirectory, підпакою, розміщеною в іншому каталозі). Постійний рух по дереву від дочірнього до батьківського елемента завжди веде до кореневого каталогу. Деякі файлові системи мають кілька корневих каталогів (наприклад, диски Windows C:\, D:\ та ін.), проте Unix та Unix-подібні операційні системи мають єдиний кореневий каталог, який називається \.

Під час роботи з файловою системою користувач завжди працює всередині деякої папки, яку називають поточною (робочою) директорією. Вивести робочу директорію можна за допомогою команди **pwd** (print working directory)

```
$ pwd
```

Вміст директорії у вигляді списку файлів та/або субдиректорій дозволяє отримати команда **ls** (list):

```
$ ls
```

Показати приховані (“dot”) файли можна за допомогою команди **ls -a**

Показати детальну інформацію щодо файлів дозволяє команда **ls -l**

Допускається комбінування кількох прапорців, на зразок **ls -l -a**

Інколи можна визначати прапорці ланцюжком: **ls -la** замість **ls -l -a**

*Якщо поточний каталог не вміщає в собі підкаталогів, створіть їх за допомогою команди **mkdir***

`$mkdir назвакаталогу`

Змінити директорію можна за допомогою команди **cd** (change directory). *Перегляньте вміст поточної директорії та перейдіть в одну з субдиректорій. Виведіть поточний каталог, потім – його вміст та ще раз. Якщо підкаталог існує, перейдіть у нього. Інакше – створіть за допомогою команди **mkdir**. Поверніться в батьківську директорію за допомогою команди*

```
$ cd ..
```

*Перейдіть до домашньої директорії, використовуючи команду **cd ~**. Перейдіть у наблищу субдиректорію та спробуйте повернутись у домашній каталог шляхом поступового підйому по дереву каталогів: **cd ../..***

За потреби послідовного виконання кількох команд можна використовувати символ «;»:

```
$ ls; pwd
```

Іншим корисним інструментом для постановки команд у ланцюжок є **&&**: наступна команда після **&&** не буде виконуватись, якщо попередня не змогла запуснитись:

```
$ cd dir1/dir2 && pwd && ls && cd
```

Запустіть дану команду в поточному вигляді та з реальними субдиректоріями на Вашій машині. Порівняйте роботу цих команд з відповідним записом через «;».

Інший символ - **&** - виглядає подібно до **&&**, проте здійснює абсолютно іншу роботу. Зазвичай виконання ланцюжка команд очікує, поки всі вони не будуть виконані, щоб дати користувачу подальшу можливість працювати з консоллю. За умови застосування **&** замість **&&** нова команда не очікуватиме завершення роботи попередніх:

```
$ cd dir1/dir2 && mvn package & cd
```

З метою виведення допомоги щодо використання деякої команди спробуйте використовувати прапорець **-h** або **--help**

```
$ du --help
```

Інформаційну довідку по команді можна отримати за допомогою команди **man** (manual). Вихід з режиму довідки здійснюється при натисненні клавіші **q**.

```
$ man ls
```

Окремі команди передбачені для перегляду та редагування файлів. Спочатку розглянемо найпростіші можливості зі створення та видалення файлів та директорій. Команда **touch** спочатку була створена для редагування таймштампу файлів, проте також може застосовуватись для створення порожнього файлу. Виконати елементарне редагування файлу можна в простому текстовому редакторі **nano**.

```
$ nano назвафайлу
```

Перехід між режимами редагування тексту та командним рядком відбувається при натисненні **Ctrl+Z** (вихід з nano) та виклику команди **fg** (повернення в nano). Створіть кілька файлів, у кожному з яких буде 10-15 рядків тексту.

У командному рядку **bash** можна запустити й інші текстові редактори, як з графічним інтерфейсом, так і без: **nedit**, **emacs**, **vi**, **vim**, **gedit**, **Notepad++**, **Atom** та ін. Сучасні текстові редактори пропонують базові зручності на зразок пошуку та заміни, підсвітки синтаксису тощо.

Наприклад, команда **head** виводить перші кілька рядків вмісту файлу. Прапорець **-n** визначає кількість таких рядків, за умовчанням їх 10:

```
$ head -n 3 назвафайлу
```

Останні декілька рядків дозволяє вивести команда **tail**. Для встановлення кількості використовуються прапорці **-n +N**, де **N** – кількість рядків:

```
$ tail -n +4 назвафайлу
```

Команда **cat** конкатенує список файлів та направляє їх у стандартний потік виводу (зазвичай у термінал). Дана команда може застосовуватись як для одного файлу, так і для кількох. Часто вона дозволяє швидко переглянути кілька файлів (при цьому може з'явитись звинувачення [UUOC](#)).

```
$ cat назвафайлу1 назвафайлу2
```

Інший інструмент швидкого перегляду файлу – команда [less](#). Вона відкриває vim-подібне вікно, текст доступний тільки для зчитування (також існує команда `more`, проте вона має менше можливостей).

Видалення файлу можна здійснити за допомогою команди **rm**, проте будьте обережні, оскільки відновленню такі файли вже не підлягають:

```
$ rm назвафайлу && ls
```

Безпечніше додати перевірочне сповіщення "are you sure?" за допомогою прапорця **-i**. Виглядає це приблизно так:

```
$ rm -i назвафайлу
```

```
rm: remove regular empty file 'назвафайлу'? y
```

Виведіть вміст директорії з файлами, видаліть один зі створених файлів та виведіть оновлений вміст директорії.

Для створення та видалення каталогів застосовують команди **mkdir** та **rmdir** відповідно. Команда **rmdir** видаляє тільки порожні директорії, проте можливо видалити каталог з усім вмістом:

```
$ rm -rf назвапапки
```

-r = recursive, -f = force. Проведіть аналогічні операції з каталогом та виведіть вміст батьківської директорії до і після видалення..

Команда переміщення файлу суміщена з командою перейменування – **mv**. Можливо перемістити файл у нову директорію без зміни назви або встановити «новий файл»:

```
$ ls && mv шляхдофайлу новийшляхдофайлу && ls
```

*Перемістіть один з файлів на рівень вище та перейменуйте його довільним чином. Поверніться до старого каталогу та скопіюйте з нього файл у новостворений каталог (команди **mkdir** та **cp**). Аргументи копіювання подібні до аргументів **mv**.*

За умовчанням, команда **mkdir** створює одну директорію. Це означає, що створити каталог **dir1/dir2/dir3** неможливо, якщо немає каталогу **dir1/dir2**. Проте за допомогою прапорця **-p** проблема вирішується:

```
$ mkdir -p dir1/dir2/dir3 && ls
```

*Спробуйте створити подібні вкладені каталоги без прапорця **-p** (зафіксувати помилку) та з ним. Візуалізуйте дерево каталогів за допомогою команди **tree**:*

```
$ tree -L 2
```

*Також продемонструйте рівень вкладеності (L) 3 для кореневого каталогу. Приховуйте порожні директорії у виводі за допомогою ключа **--prune***

```
$ tree --prune
```

Зауважте, що також з виводу прибираються «рекурсивно порожні» директорії, тобто каталоги, в яких присутні підкаталоги, проте всі вони всередині порожні.

Наступним кроком є оцінка зайнятості дискового простору наявними файлами. Може використовуватись команда **df**, яка дозволяє визначати, скільки місця займають файли на дисках системи (жорстких дисках та ін.):

```
$ df -h
```

Прапорець **-h** у даному контексті означає "human-readable". Деякі команди використовують цю угоду для відображення розмірів файлів та дисків з позначками К (кілобайти), G (гігабайти) тощо замість виводу величезних чисел для представлення кількості байтів.

Команда **du** показує використання простору пам'яті конкретною директорією або її субдиректоріями. Опційний параметр **--max-depth=N** обмежує показ директорій в N рівнів:

```
$ du -h --max-depth=1
```

Командний рядок також має в розпорядженні команду для показу всіх працюючих у даний момент процесів (jobs):

```
$ ps
```

Змініть пароль облікового запису на більш захищений! Використовуйте команду `passwd`:

```
$ passwd
Changing password for назвааакаунта.
(current) UNIX password: <type current password>
Enter new UNIX password: <type new password>
Retype new UNIX password: <type new password again>
passwd: password updated successfully
```

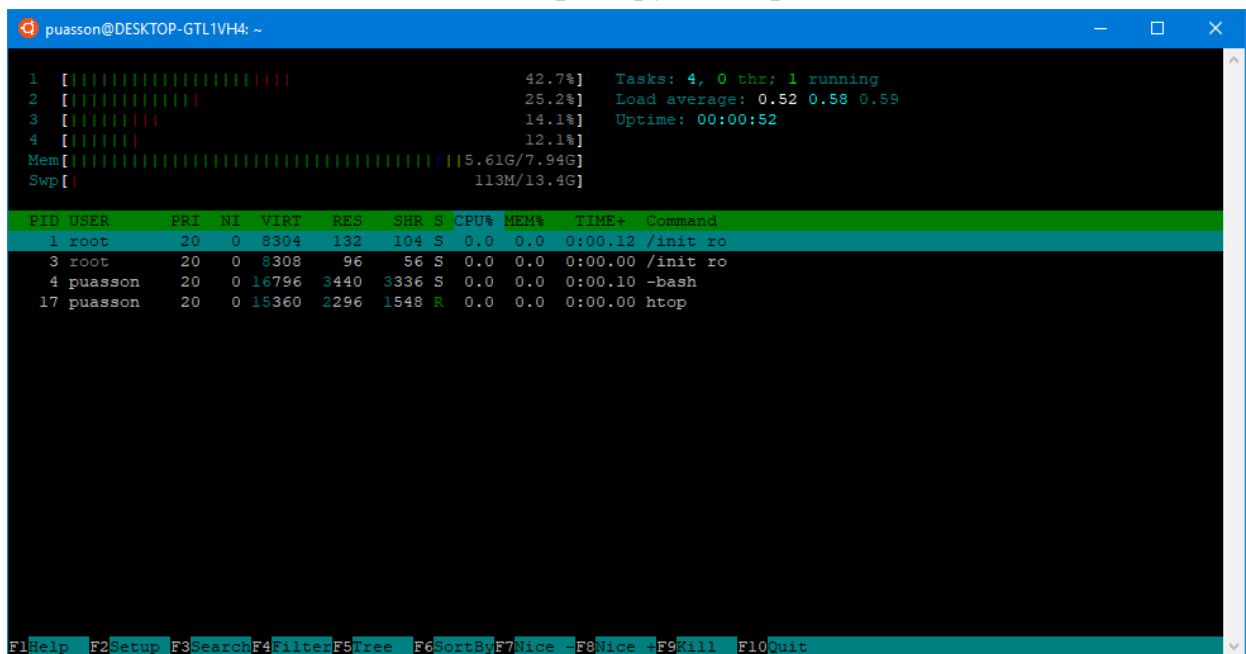
Почистити екран від тексту можна за допомогою команди **clear**, яка просто додасть достатню кількість переходів на новий рядок.

3. Додаткові можливості командного рядка `bash`

Детальніший огляд файлового простору можна отримати за допомогою команди **ncdu** (NCurses Disk Usage). Вона відкриває vim-подібне вікно (q для виходу), яке доступне лише для читання:

```
$ ncdu
```

Усі запущені в даний момент процеси та їх власників, займану пам'ять та ін. можна відобразити, застосувавши команду **top**. Покращена, інтерактивна версія викликається за допомогою команди **htop**. Можна додати прапорець **-u назвакористувача**, щоб відфільтрувати процеси за їх власником.



```
puasson@DESKTOP-GTL1VH4: ~
1  [|||||] 42.7% Tasks: 4, 0 thr; 1 running
2  [|||||] 25.2% Load average: 0.52 0.58 0.59
3  [|||||] 14.1% Uptime: 00:00:52
4  [|||||] 12.1%
Mem[|||||] 5.61G/7.94G
Swp[|] 113M/13.4G

PID USER PRI NI VIRT RES SHR S CPU% MEM% TIME+ Command
1 root 20 0 8304 132 104 S 0.0 0.0 0:00.12 /init ro
3 root 20 0 8308 96 56 S 0.0 0.0 0:00.00 /init ro
4 puasson 20 0 16796 3440 3336 S 0.0 0.0 0:00.10 -bash
17 puasson 20 0 15360 2296 1548 R 0.0 0.0 0:00.00 htop

F1Help F2Setup F3Search F4Filter F5Tree F6SortBy F7Nice -F8Nice -F9Kill F10Quit
```

За допомогою `bash` можна викликати REPL (Read-Evaluate-Print Loop) для конкретних мов програмування. Наприклад, консоль мови Python запускається командою **python**, **python3** тощо. Виклик `quit()` або `Ctrl+D` дозволить вийти з даного режиму.

```
puasson@DESKTOP-GTL1VH4: ~  
puasson@DESKTOP-GTL1VH4:~$ python3  
Python 3.6.7 (default, Oct 22 2018, 11:32:17)  
[GCC 8.2.0] on linux  
Type "help", "copyright", "credits" or "license" for more information.  
>>> quit()  
puasson@DESKTOP-GTL1VH4:~$
```

Аналогічні дії доступні для мови Java за допомогою команди **jshell**. У випадку відсутності потрібних модулів командний рядок запропонує довстановити їх.

Більшість команд чи програм мають корисний прапорець **-version** або **--version**, який показує їх версію. Зверніть увагу, що деякі програми використовують прапорець `-v` для показу версії, а інші – для управління деталізацією виводу інформації (verbosity).

```
$ ls -version  
$ ncdu -version  
$ python3 --version
```

Командний рядок дозволяє вивести змінні середовища, доступні для операційної системи:

```
$ printenv
```

Проте також можна створювати власні змінні та присвоювати їм значення. Вивести таке значення дозволяє команда **echo**. За наявності пробілів значення слід огорнути подвійними лапками. Дані змінні буде видно в межах працюючого процесу, проте за потреби розширення видимості на підпроцеси слід дописати команду **export**:

```
myvar=hello  
$ echo $myvar  
$ myvar="hello, world!" && echo $myvar  
$ export myvar="another one" && echo $myvar
```

Допускається видалення значення змінної шляхом застосування команди **unset** або присвоєнням порожнього значення (відсутністю тексту після знаку «=»). Зверніть увагу, що для виведення значення командою `echo` слід дописувати символ `$` перед назвою змінної.

```
$ unset mynewvar
```

```
puasson@DESKTOP-GTL1VH4: ~  
puasson@DESKTOP-GTL1VH4:~$ myvar=hello  
puasson@DESKTOP-GTL1VH4:~$ myvar=  
puasson@DESKTOP-GTL1VH4:~$ echo myvar  
myvar  
puasson@DESKTOP-GTL1VH4:~$ echo $myvar  
puasson@DESKTOP-GTL1VH4:~$ _
```

Додаткові зручності пропонує запис довгих команд у змінні-псевдоніми (aliases), що в подальшому може спростити роботу з командним рядком. Для цього слід записати команду в форматі

alias назвазмінної="тексткоманди"

```
puasson@DESKTOP-GTL1VH4: ~  
puasson@DESKTOP-GTL1VH4:~$ alias lc="ls -l -a -h -t"  
puasson@DESKTOP-GTL1VH4:~$ lc  
total 8.0K  
drwxr-xr-x 1 puasson puasson 4.0K Feb 16 10:01 .  
-rw----- 1 puasson puasson  16 Feb 16 10:01 .python_history  
drwx----- 1 puasson puasson 4.0K Feb 16 09:50 .config  
-rw----- 1 puasson puasson  235 Feb 14 07:24 .bash_history  
-rw-r--r-- 1 puasson puasson    0 Feb 13 08:38 .sudo_as_admin_successful  
drwxrwxrwx 1 puasson puasson 4.0K Feb 13 08:20 test2  
drwxrwxrwx 1 puasson puasson 4.0K Feb 13 08:09 test1  
-rw-r--r-- 1 puasson puasson  807 Feb 13 07:57 .profile  
-rw-r--r-- 1 puasson puasson 3.7K Feb 13 07:57 .bashrc  
-rw-r--r-- 1 puasson puasson  220 Feb 13 07:57 .bash_logout  
drwxr-xr-x 1 root    root    4.0K Feb 13 07:57 ..  
puasson@DESKTOP-GTL1VH4:~$
```

Видалити псевдонім дозволяє команда **unalias**:

```
$ unalias lc
```

Про відмінності між змінними та псевдонімами можна почитати [тут](#). Деякі програми, на зразок Git, дозволяють створювати псевдоніми тільки для використання в цих же програмах. Загалом командний рядок bash має власну просту скриптову мову, роботу з якою будемо розглядати в наступному практичному занятті.

Командний рядок має широкі можливості з пошуку. Команда **whereis** шукає «ймовірно корисні» файли, пов'язані з конкретною командою:

```
$ whereis ls
```

За допомогою команди **which** можна отримати розташування бінарного файлу шуканої команди. Первинна версія команди, яка прихована за псевдонімом, також може бути визначена командою **which**:


```
$ which ls
```

```
/bin/ls
```

```
puasson@DESKTOP-GTL1VH4: ~
```

```
puasson@DESKTOP-GTL1VH4:~$ alias ls="ls -l"
```

```
puasson@DESKTOP-GTL1VH4:~$ ls
```

```
total 0
```

```
drwxrwxrwx 1 puasson puasson 4096 Feb 13 08:09 test1
```

```
drwxrwxrwx 1 puasson puasson 4096 Feb 13 08:20 test2
```

```
puasson@DESKTOP-GTL1VH4:~$ /bin/ls
```

```
test1 test2
```

```
puasson@DESKTOP-GTL1VH4:~$
```

Також присутня команда **whatis**, яка виводить однорядковий опис команд з їх man-сторінки:

```
$ whatis whereis which whatis
```

```
whereis (1)          - locate the binary, source, and manual page files for a command
```

```
which (1)            - locate a command
```

```
whatis (1)           - display one-line manual page descriptions
```

Шукати файл в системі допомагає команда **locate**. Вона звертається до періодично оновлюваного кешованого списку файлів. Оскільки пошук іде лише по списку, така реалізація швидша за альтернативну команду – **find**.

```
puasson@DESKTOP-GTL1VH4: ~
```

```
puasson@DESKTOP-GTL1VH4:~$ locate README.md
```

```
/usr/share/doc/bash/README.md.bash_completion.gz
```

```
/usr/share/doc/bash-completion/README.md.gz
```

```
/usr/share/doc/command-not-found/README.md
```

```
/usr/share/doc/git/README.md
```

```
/usr/share/doc/iso-codes/README.md.gz
```

```
/usr/share/doc/libfuse2/README.md
```

```
/usr/share/doc/libgeoip1/README.md.gz
```

```
/usr/share/doc/libgnutls30/README.md.gz
```

```
/usr/share/doc/libidn2-0/README.md.gz
```

```
/usr/share/doc/networkd-dispatcher/README.md.gz
```

```
/usr/share/doc/psmisc/README.md
```

```
/usr/share/doc/python3-httpplib2/README.md
```

```
/usr/share/doc/python3-json-pointer/README.md
```

```
/usr/share/doc/python3-jsonpatch/README.md
```

```
/usr/share/doc/sosreport/README.md
```

```
/usr/share/doc/tcpdump/README.md.gz
```

```
/usr/share/doc/unattended-upgrades/README.md.gz
```

```
puasson@DESKTOP-GTL1VH4:~$
```

Команда **find** проходить по файловій системі, шукаючи файли, які існують у даний момент. Для команди `locate` це може бути не так. Крім того, перша версія `find` була написана ще в 1971 році для Unix, тому є набагато більш поширеною, ніж команда `locate`, що з'явилась у 1994 році в GNU. Також команда `find` має набагато більше можливостей: може шукати файли за тривалістю їх зберігання, розміром, власником, типом, дозволами, глибиною вкладеності, часовому штампі; застосовувати регулярні вирази для пошуку та ін.

```
puasson@DESKTOP-GTL1VH4: ~  
puasson@DESKTOP-GTL1VH4:~$ find /usr/ -iname "README.md"  
/usr/share/doc/command-not-found/README.md  
/usr/share/doc/git/README.md  
/usr/share/doc/libfuse2/README.md  
/usr/share/doc/psmisc/README.md  
/usr/share/doc/python3-httpplib2/README.md  
/usr/share/doc/python3-json-pointer/README.md  
/usr/share/doc/python3-jsonpatch/README.md  
/usr/share/doc/sosreport/README.md  
/usr/share/doc/wslu/README.md  
puasson@DESKTOP-GTL1VH4:~$
```

Командний рядок `bash` має кілька команд для обміну даними в мережі. Наприклад, команда **ping** намагається відкрити лінію комунікації з хостом у мережі. Зазвичай вона використовується для перевірки наявності Інтернет-з'єднання та переривається комбінацією клавіш `Ctrl+C`:

```
$ ping google.com
```

Для простого завантаження файлів з мережі Інтернет можна використовувати команди **wget** або **curl**:

```
$ wget \  
> http://releases.ubuntu.com/19.10/ubuntu-19.10-desktop-amd64.iso  
  
$ curl \  
> http://releases.ubuntu.com/19.10/ubuntu-19.10-desktop-amd64.iso \  
> --output ubuntu.iso
```

Команда **curl** може застосовуватись у такій же манері, як і `wget` (не забувайте прапорець `--output`). Обидві команди мають свої сильні та слабкі сторони. `curl` підтримує набагато більше протоколів та ширше розповсюджена, ніж `wget`; також `curl` може надсилати дані, а `wget` – тільки отримувати їх. У той же час, `wget` дозволяє завантажувати файли рекурсивно, що не підтримується командою `curl`.

Розроблені на базі Debian дистрибутиви Linux мають чудовий інструмент управління пакетами **apt**. Він використовується для інсталяції, оновлення чи видалення програмного забезпечення на машині. Для пошуку конкретного програмного засобу використовуйте команду **apt search**, а для встановлення — **apt install**:

```
$ apt search bleachbit
```

```
$ sudo apt install bleachbit
```

Програмне забезпечення для Linux часто надходить запакованим у файли **.tar.gz** ("tarball"):

```
$ wget \
> https://github.com/atom/atom/releases/download/v1.35.0-beta0/atom-amd64.tar.gz
```

Такі файли можна розпакувати за допомогою команди **gunzip**:

```
$ gunzip atom-amd64.tar.gz && ls
```

Файл **.tar.gz** розпакується в файл **.tar**, який надалі можна розархівувати в папку з файлами командою **tar -xf** (-x означає "extract", -f задає файл):

```
$ tar -xf atom-amd64.tar && mv \
```

Для обернених операцій можна створити (-c) tar-файл з папки та заархівувати (або розархівувати) його з прапорцем -z:

```
$ tar -zcf compressed.tar.gz atom && ls
```

.tar-файли також можна заархівувати за допомогою команди **gzip**:

```
$ gzip atom-amd64.tar && ls
```

За умовчанням shell-команди зчитують введену інформацію зі стандартного потоку вводу (stdin або 0) та записують в стандартний потік виводу (stdout або 1), поки не трапиться помилка, яка спрямує вивід у стандартний потік помилок (stderr або 2).

Команда **echo** записує текст у stdout за умовчанням, що в більшості ситуацій призводить до друку інформації в консоль (термінал):

```
$ echo "hello"
```

Конвеєрний (pipe) оператор **|** перенаправляє вивід першої команди на ввід другої:

```
$ echo "example document" | wc
```

> перенаправляє вивід з stdout у конкретне місце:

```
$ echo "test" > file && head file
```

Команда **printf** є покращеною версією **echo**, дозволяє формувати та екранувати текст:

```
$ printf "1\n3\n2"
```

```
1
3
2
```

← отримує ввід з деякого місця, а не stdin:

```
$ sort <(printf "1\n3\n2")
```

```
1
2
3
```

Рекомендований спосіб спрямувати вміст файлу команді – використати <. Зауважте, що це спричиняє дані «перетікати» справа наліво в командному рядку:

```
$ printf "1\n3\n2" > file && sort < file
```

```
1
2
3
```

Потоки вводу та виводу можна перенаправити за допомогою операторів |, > та <, проте stdin, stdout та stderr також можна змінювати напряму, використовуючи числові ідентифікатори. Наприклад, запис в stdout або stderr можна здійснити операторами >&1 або >&2 відповідно:

```
$ cat test
echo "stdout" >&1
echo "stderr" >&2
```

За умовчанням, stdout та stderr виводять дані в термінал:

```
$ ./test

stderr
stdout
```

Перенаправлення stdout в /dev/null залишає для виводу в консоль лише stderr:

```
./test 1>/dev/null

stderr
```

Аналогічна операція з stderr призводить до виводу в термінал stdout:

```
$ ./test 2>/dev/null
```

```
stdout
```

Перенаправлення всього виводу в /dev/null:

```
$ ./test &>/dev/null
```

Спрямувати вивід в stdout та довільну кількість інших місць можна за допомогою команди **tee**:

```
ls && echo "test" | tee file1 file2 file3 && ls  
  
file0  
test  
file0 file1 file2 file3
```

4. Розширені можливості командного рядка bash

У командному рядку присутні команди для управління обліковими записами та групами. Враховуючи, що користувачі можуть логінитись багато разів, коли, наприклад, підключається по ssh-протоколу, отримати повний перелік можна з файлу /etc/passwd. **ОБЕРЕЖНО!** Не вносьте зміни в цей файл, оскільки це може пошкодити аккаунти користувачів! Детальніше [тут](#).

```
puasson@DESKTOP-GTL1VH4: ~  
puasson@DESKTOP-GTL1VH4:~$ alias au="cut -d: -f1 /etc/passwd | sort | uniq"  
puasson@DESKTOP-GTL1VH4:~$ au  
apt  
backup  
bin  
daemon  
dnsmasq  
games  
gnats  
irc  
landscape  
list  
lp  
lxd  
mail  
man  
messagebus  
news  
nobody  
pollinate  
proxy  
puasson  
root  
sshd  
sync  
sys  
syslog  
systemd-network  
systemd-resolve  
testuser
```

У переліку користувачів можете побачити, зокрема, обліковий запис суперкористувача (root), поточний обліковий запис (тут – puasson), а також аккаунт testuser, який перед цим було додано за допомогою команди **useradd**:

```
$ sudo useradd testuser
```

Командний рядок дозволяє перевіряти назву активного облікового запису за допомогою команди **whoami**:

```
$ whoami
```

та запускати команди з-під іншого користувача шляхом виклику

sudo -u назвакористувача (потрібно буде ввести пароль):

```
puasson@DESKTOP-GTL1VH4: ~  
puasson@DESKTOP-GTL1VH4:~$ sudo -u testuser touch def && ls -l  
[sudo] password for puasson:  
touch: cannot touch 'def': Permission denied  
puasson@DESKTOP-GTL1VH4:~$
```

Якщо не дописати -u, вважатиметься, що команда виконуватиметься суперадміністратором (обліковий запис root). У даному випадку виконати команду створення файлу з назвою def неможливо, оскільки немає необхідних дозволів для цього для аккаунта testuser. Крім того, на даному етапі обліковий запис не активований, оскільки не встановлено пароль для його подальшого використання.

```
puasson@DESKTOP-GTL1VH4: ~  
puasson@DESKTOP-GTL1VH4:~$ passwd testuser  
passwd: You may not view or modify password information for testuser.  
puasson@DESKTOP-GTL1VH4:~$ sudo passwd testuser  
[sudo] password for puasson:  
Enter new UNIX password:  
Retype new UNIX password:  
passwd: password updated successfully
```

Відкрийте ще одну консоль та переключіться в інший аккаунт за допомогою команди **su**:

```
puasson@DESKTOP-GTL1VH4: ~  
puasson@DESKTOP-GTL1VH4:~$ su testuser  
Password:  
$ whoami  
testuser  
$
```

Таким чином, матимемо дві консолі, які є робочим простором для відповідних облікових записів. Зробимо облікові записи рівноправними, додавши новостворений аккаунт у групу sudo (адміністративних облікових записів):

```
puasson@DESKTOP-GTL1VH4: ~  
puasson@DESKTOP-GTL1VH4:~$ sudo usermod -aG sudo testuser  
[sudo] password for puasson:
```

Аналогічні дії можна виконувати з групами користувачів: команди [groupmod](#)

```
puasson@DESKTOP-GTL1VH4: ~  
puasson@DESKTOP-GTL1VH4:~$ alias ag="cut -d: -f1 /etc/group | sort"  
puasson@DESKTOP-GTL1VH4:~$ ag  
adm  
admin  
audio  
backup  
bin  
cdrom  
crontab  
daemon  
dialout  
dip  
disk  
fax  
floppy  
games  
gnats  
input  
irc  
kmem  
landscape  
list  
lp  
lxd  
mail  
man  
messagebus  
mlocate  
netdev
```

Команда `su` дозволяє тимчасово перейти до іншого облікового запису. Для повернення до первинного аккаунту слід набрати команду **exit**. Детальніше про відмінності команд `sudo` і `su` можете розглянути [тут](#).

Суперкористувач (зазвичай `root`) – єдиний аккаунт, з якого можна встановлювати програмне забезпечення, створювати користувачів та ін. Якщо про це забути, можна отримати подібну помилку:

```
$ apt install ruby  
  
E: Could not open lock file /var/lib/dpkg/lock-frontent - open (13: Permission denied)  
E: Unable to acquire the dpkg frontend lock (/var/lib/dpkg/lock-frontent), are you root?
```

Допописування перед цією командою `sudo` вирішить проблему. Або можна використати шорткат «`!!`»:

```
$ apt install ruby

E: Could not open lock file /var/lib/dpkg/lock-frontent - open (13: Permission denied)
E: Unable to acquire the dpkg frontend lock (/var/lib/dpkg/lock-frontent), are you root?

[ andrew@pc01 ~ ]$ sudo !!
sudo apt install ruby
Reading package lists...
```

Правильний ввід паролю при запуску `sudo`-команд зазвичай дає 15 хвилин для роботи в режимі суперкористувача. Після їх завершення буде потрібно вводити команду `sudo` заново.

Працюючи з файлами, можна зчитувати (r), записувати (w) та/або виконувати (x) їх від різних користувачів, груп користувачів або й без них. Дозволи для файлу можна переглянути за допомогою команди **`ls -l`**. Результат представляється 10-символьним рядком, наприклад, **`drwxr-xr-x`**.

```
$ ls -lh

total 8
drwxr-xr-x 4 andrew andrew 4.0K Jan  4 19:37 tast
-rwxr-xr-x 1 andrew andrew  40 Jan 11 16:16 test
-rw-r--r-- 1 andrew andrew   0 Jan 11 16:34 tist
```

Перший символ кожного рядка представляє тип файлу (d = директорія, l = посилання (лінк), - = звичайний файл тощо); потім представлено три групи символів, які описують дозволи, встановлені користувачем-власником (u), групою-власницею (g) та іншими користувачами (o). Число після рядка вказує кількість посилань у файловій системі на цей файл.

- r – дозвіл на читання (read),
- w – дозвіл на запис (write),
- x – можливість виконання (executable).

Якщо директорія має дозвіл x, це значить, що її можна відкрити, а вміст – перелічити. Ці три дозволи зазвичай представляються одним десятковим числом, отриманим з 3-бітового двійкового числа (1 – дозволено, 0 – заборонено). Наприклад, r-x -> 101 -> 5. Тому попередні файли мають дозволи 755, 755 та 644 відповідно.

Рядки продовжуються ім'ям власника та його групою. Потім виводиться розмір файлу, час останніх змін та назва файлу. Прапорець `-h` робить вивід читабельним.

Внести зміни до дозволів для файлу можна за допомогою команди **chmod**:

```
$ chmod 777 test && chmod 000 tist && ls -lh

total 8.0K
drwxr-xr-x 4 andrew andrew 4.0K Jan  4 19:37 tast
-rwxrwxrwx 1 andrew andrew  40 Jan 11 16:16 test
----- 1 andrew andrew    0 Jan 11 16:34 tist
```

або додаючи (+) або видаляючи (-) дозволи `gwx` за допомогою прапорців:

```
$ chmod +rwx tist && chmod -w test && ls -lh

chmod: test: new permissions are r-xrwxrwx, not r-xr-xr-x
total 8.0K
drwxr-xr-x 4 andrew andrew 4.0K Jan  4 19:37 tast
-r-xrwxrwx 1 andrew andrew  40 Jan 11 16:16 test
-rwxr-xr-x 1 andrew andrew   0 Jan 11 16:34 tist
```

Власника файлу можна змінити за допомогою команди **chown**

```
$ sudo chown назвааккаунту test
```

Відповідну групу можна замінити командою **chgrp**:

```
$ sudo chgrp hadoop tist && ls -lh

total 8.0K
drwxr-xr-x 4 andrew andrew 4.0K Jan  4 19:37 tast
-----w--w- 1 marina andrew  40 Jan 11 16:16 test
-rwxr-xr-x 1 andrew hadoop    0 Jan 11 16:34 tist
```

Командний рядок `bash` має багато функціональних можливостей для роботи з текстом. Команда `uniq` дозволяє друкувати унікальні (за умовчанням) або повторювані рядки. Наприклад, виконає запис у два файли:

```
$ printf "1\n2\n2" > файл1 && \> printf "1\n3\n2" > файл2
```

а потім викличемо команду

```
$ uniq файл1
```

Команда `sort` дозволить відсортувати рядки в алфавітно-цифровому порядку:

```
$ sort файл2
```

```
1  
2  
3
```

Команда **diff** повідомить про те, які рядки відрізняються між двома файлами:

```
$ cmp файл1 файл2
```

```
файл1 файл2 differ: char 3, line 2
```

Команда **cmp** побайтово порівнює файли та виводить відмінності:

```
$ cmp файл1 файл2
```

```
файл1 файл2 differ: char 3, line 2
```

Також доступна команда **cut**, призначена для розрізання рядка на секції, відокремлені символом-роздільником (корисно для обробки CSV-файлів). Прапорець **-d** задає роздільник, а **-f** - field index для друку (починаючи з 1 для першого поля):

```
$ printf "137.99.234.23" > файл3
```

```
$ cut -d'.' файл3 -f1
```

```
137
```

Команда **sed** широко використовується для заміни одного рядка іншим всередині файлу:

```
$ echo "old" | sed s/old/new/
```

Проте **sed** – дуже потужна утиліта. Вона дозволяє виконувати заміни на базі регулярних виразів, вибірково друкувати рядки файлу, які відповідають або містять деякий паттерн, редагувати текстові файли на льоту або неінтерактивно та багато іншого. Кілька хороших туторіалів по **sed**: [тут](#), [тут](#) і [тут](#).

Командний рядок **bash** пропонує набір операцій на базі зіставлення паттернів. Наприклад, команда **grep** (globally for a regular expression and print it) використовується для пошуку тексту в файлах.

```
$ grep -e ".*fi.*" /etc/profile
```

або

```
$ grep "назвааккаунту" /etc/passwd
```

Дана команда – хороший вибір для простого пошуку рядків у файлі, якщо планується дозволяти іншим програмам працювати з цими рядками (або тільки переглядати їх). `grep` підтримує прапорець `-E` для розширених регулярних виразів, `-F` для зіставлення з будь-яким одним рядком з багатьох за раз, `-r` рекурсивно шукає файли всередині папки. Ці прапорці використовуються для реалізації як окремі команди: `egrep`, `fgrep` та `rgrep`, проте зараз вони вважаються застарілими.

Команда **`awk`** підтримує мову зіставлення за шаблоном, побудовану навколо зчитування та оперування делімітованими файлами даних, на зразок CSV. Для орієнтування: `grep` доречна для знаходження рядків та шаблонів у файлах, `sed` – для заміни рядків у файлах виду 1:1, а `awk` – для виділення рядків та шаблонів з файлів для подальшого аналізу. Для прикладу роботи з командою `awk` розглянемо файл з двома стовпцями даних

```
$ printf "A 10\nB 20\nC 60" > file
```

Виконаємо цикл з ітераціями по рядках, додаватимемо числа до суми, інкрементуватимемо лічильник та виведемо середнє значення:

```
$ awk 'BEGIN {sum=0; count=0; OFS=" "} {sum+=$2; count++} END {print "Average:", sum/count}' file
```

Вивід: Average: 30. Детальніше про відмінності між `sed`, `grep`, `awk` можна розглянути [тут](#).

Наступні команди, починаючи з розділу «Copying Files Over ssh» перечитайте для саморозвитку [тут](#).


```
puasson@DESKTOP-GTL1VH4: ~  
puasson@DESKTOP-GTL1VH4:~$ sudo touch ghi && ls -l  
total 0  
-rw-r--r-- 1 root    root      0 Feb 16 11:58 def  
-rw-r--r-- 1 root    root      0 Feb 16 11:59 ghi  
drwxrwxrwx 1 puasson puasson 4096 Feb 13 08:09 test1  
drwxrwxrwx 1 puasson puasson 4096 Feb 13 08:20 test2  
puasson@DESKTOP-GTL1VH4:~$
```

```
puasson@DESKTOP-GTL1VH4: ~  
puasson@DESKTOP-GTL1VH4:~$ sudo usermod -aG testgroup testuser  
puasson@DESKTOP-GTL1VH4:~$ groups  
puasson adm dialout cdrom floppy sudo audio dip video plugdev lxd netdev  
puasson@DESKTOP-GTL1VH4:~$
```

Далі буде: Advanced

<https://dev.to/awwsmm/101-bash-commands-and-tips-for-beginners-to-experts-30je>