Міністерство освіти і науки України

Національний авіаційний університет

Факультет кібербезпеки, комп’ютерної та програмної інженерії

Кафедра інженерії програмного забезпечення

*Операційні системи*

**Лабораторна робота №8**

**«Можливості командної оболонки»**

Розробив: студент ПІ-322б

Царук С.О.

Перевірив:

Лосєв Є.О.

Київ 2021

**Мета:** Визначення понять і аналіз роботи нижче названих об’єктів та процесів, які відбуваються при доступі процесів до файлів і каталогів.

**Завдання**

Опрацювання роботи команд:

1. Налаштування термінальної лінії.
2. Екранування символів, які є управляючими.
3. Пошук по історії команд.
4. Перегляд раніше визначених скорочень.
5. Використання скорочень і pushd/popd.
6. Використання добудови.
7. Використання шаблону в командному рядку.
8. Екранування спеціальних символів і обробка «пустих» шаблонів.
9. Використання утиліти date у змінній оточення LC\_TIME.
10. Підставлення значень змінних.
11. Експорт змінних shell в оточення.
12. Використання PATH.
13. Використання змінної оточення PS1.
14. Використання оператора if який використовує код повернення програми grep.
15. Використання оператора test.
16. Підставлення виводу команди.
17. Використання позиційних параметрів в сценарії.
18. Використання for і операції «.»

Зміст

[Хід роботи 4](#_Toc67499666)

[1. Налаштування термінальної лінії 4](#_Toc67499667)

[2. Екранування символів, які є управляючими 4](#_Toc67499668)

[3. Пошук по історії команд 4](#_Toc67499669)

[4. Перегляд раніше визначених скорочень 5](#_Toc67499670)

[5. Використання скорочень і pushd/popd 5](#_Toc67499671)

[6. Використання добудови 6](#_Toc67499672)

[7. Використання шаблону в командному рядку 7](#_Toc67499673)

[8. Екранування спеціальних символів і обробка «пустих» шаблонів 7](#_Toc67499674)

[9. Використання утиліти date у змінній оточення LC\_TIME 8](#_Toc67499675)

[10. Підставлення значень змінних 8](#_Toc67499676)

[11. Експорт змінних shell в оточення 9](#_Toc67499677)

[12. Використання PATH 9](#_Toc67499678)

[13. Використання змінної оточення PS1 10](#_Toc67499679)

[14. Використання оператора if який використовує код повернення програми grep 11](#_Toc67499680)

[15. Використання оператора test 11](#_Toc67499681)

[16. Підставлення виводу команди 12](#_Toc67499682)

[17. Використання позиційних параметрів в сценарії 12](#_Toc67499683)

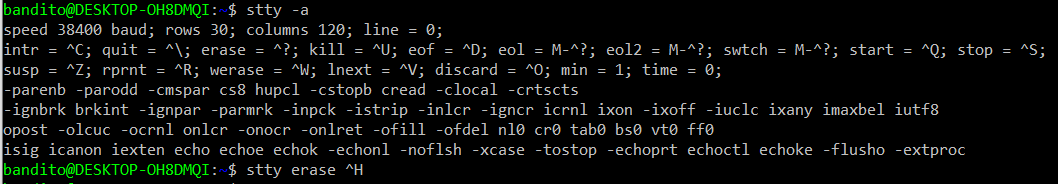
[18. Використання for і операції «.» 13](#_Toc67499684)

[Висновок 15](#_Toc67499685)

# Хід роботи

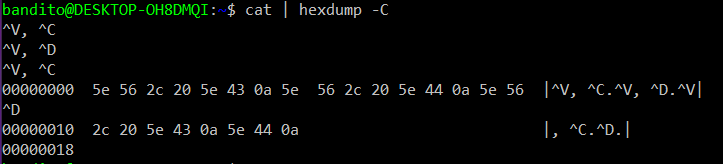
## Налаштування термінальної лінії

В дійсності, усі управляючі символи, які інтерпретує система, можна пере налаштовувати за допомогою команди ***stty***. Повний список того, що можна налаштовувати, видає команда ***stty –а.***



## Екранування символів, які є управляючими

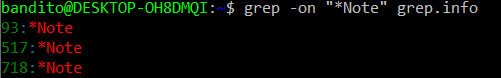
На кінець, щоби *позбавити* символ, який передається, управляючих функцій (якщо, наприклад, вимагається передати програмі на введення символ з кодом ***3***, тобто «***^C***»), безпосередньо, перед введенням цього символу, треба подати команду «***^V***» (***lnext***):



## Пошук по історії команд

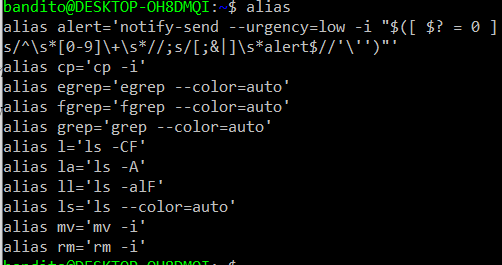
Якщо необхідно добути з історії якусь давню команду, простіше не ганяти список історії стрілками, *пошукати* в ній за допомогою команди «***^R***» (***r****everse search*). При цьому виводиться підказка спеціального виду («(***reverse -і-search***)»), підрядок пошуку (оточений символами ***‘*** i **’**) і остання з команду історії, в якій ця підказка присутня





## Перегляд раніше визначених скорочень

Пошук по історії – зручний засіб: довгий командний рядок можна не набирати повністю, знайти і використати. Одначе, *давню* команду прийдеться добувати за допомогою декількох «***^R***» – а можна і зовсім не дошукатися, якщо вона вже вибула звідти. Для того, щоби оперативно замінювати короткі команди довгими, варто скористатися **скороченнями** (***aliases***). У конфігураційних файлах командного інтерпретатора користувача, зазвичай, *вже* визначено декілька скорочень, список яких можна переглянути за допомогою команди ***aliases*** без параметрів



## Використання скорочень і pushd/popd

Від «***-і***», яке надоїло, ми позбулися за допомогою команди ***unalias***, а за одне, ввели скорочення для **bash** – ***pushd*** і ***popd***. Ці команди, подібно ***cd***, змінюють каталог. Різниця складається в тому, що ***pushd*** всі каталоги, які користувач робить поточними, запам’ятовує в особливому списку (стеку). Команда ***popd*** знищує останній елемент цього стеку, і робить поточний каталог перед останнім.



## Використання добудови

Щоби не дописувати літери, які залишилися (а імена файлів в Linux можуть бути досить довгими), необхідно нажати клавішу *Tab*. І ***bash*** сам добудує ім’я файлу до цілого (знову скористаємося методом кадрів)



TAB





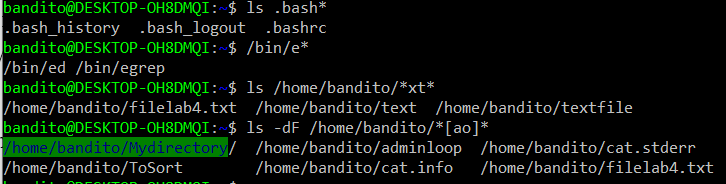
TAB



Виконуючи **добудову** (*completion*), ***bash*** може вивести не весь рядок, а тільки ту частину, відносно якої у нього немає сумнівів. Якщо подальша добудова може піти *декількома* шляхами, то одноразовий натиск *Tab*, призведе до того, що ***bash*** розгублено пискне, другий натиск приведе до виведення під командним рядком списку усіх можливих варіантів.

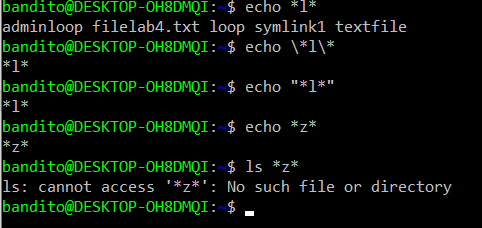
## Використання шаблону в командному рядку

Шаблони використовуються у декількох конструкціях ***shell***. Головне місце їх застосування – командний рядок. Якщо оболонка бачить в командному рядку шаблон, вона негайно замінює його на список *файлів*, імена яких йому відповідають. Команда. Яка потім викликається, отримує у якості параметрів список файлі уже без всіляких шаблонів, так якби цей список користувач ввів вручну. Ця здібність командного інтерпретатора називається ***генерацією імен файлів***.



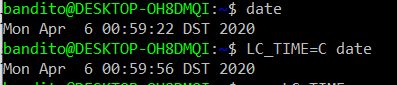
## Екранування спеціальних символів і обробка «пустих» шаблонів

Якщо, перед любим спеціальним символом стоїть «***\***», цей символ позбавляється спеціального значення, **екранується**:пара «***\\символ***» замінюється командним інтерпретатором на «***символ***» і передається у командний рядок без усіякої подальшої обробки



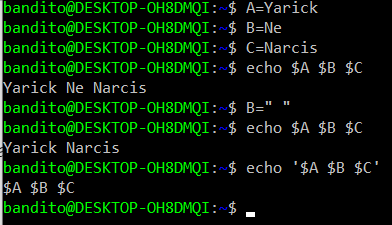
## Використання утиліти date у змінній оточення LC\_TIME

Mи привласнили деяке значення змінній оточення у командному рядку *перед* іменем команди. Командний інтерпретатор, побачивши «***=***» в середині першого слова командного рядку, приходить до висновку, що це – операція присвоєння, а не ім’я команди, і запам’ятовує, як треба змінити оточення команди, яка іде після. Змінна оточення ***LC\_TIME*** наказує використовувати визначену мову при виведенні дати і часу, а значення «***С***» відповідає «стандартній системній» мові (частіше за все - англійській).



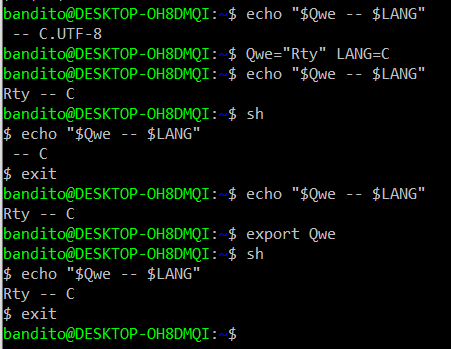
## Підставлення значень змінних

Якщо розглядати ***shell*** у якості високорівневої мови програмування, її змінні – *звичайні* рядкові змінні. Записати значення у змінну можна за допомогою операції присвоювання, а прочитати його звідти – за допомогою операції підстановки виду ***$змінна***



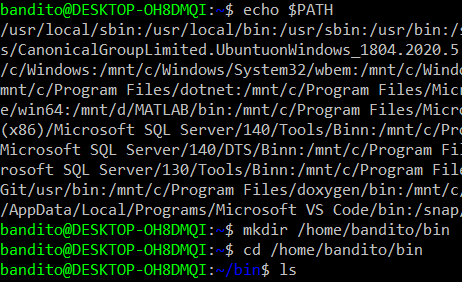
## Експорт змінних shell в оточення

Змінні, які командний інтерпретатор **bash** визначає *після* запуску, не належать оточенню, і, стало бути, не успадковуються дочірніми процесами. Щоби змінна **bash** попала в оточення, її треба *експортувати* командою ***export***



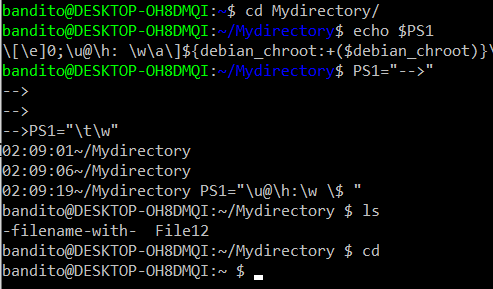
## Використання PATH

Вельми примітна змінна оточення ***PATH***. В ній утримується список каталогів, елементи якого розділяються двокрапкою. Якщо команда в командному рядку – не особиста команда ***shell*** ( на зразок ***cd***) і не подана у вигляді *шляху* до файлу, який запускається, (як ***/bin/ls*** або ***./script***), то shell буде шукати цю команду серед імен файлів, які запускаються



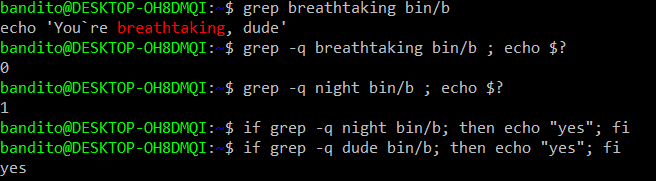
## Використання змінної оточення PS1

Деякі змінні оточення призначені спеціально для ***bash***: вони задають його властивості і особливості поведінки. Такі змінні сімейства ***PS*** (***P****rompt* ***S****tring*). У цих змінних зберігається рядок-підказка, яку командний інтерпретатор виводить урізних станах. Зокрема, вміст ***PS1*** – це підказка , яку **shell** показує, коли виводить командний рядок, а ***PS2*** – коли користувач натискає *Enter*, а інтерпретатор з якої-небудь причини вважає, що введення командного рядку не завершено (наприклад. не закриті лапки).



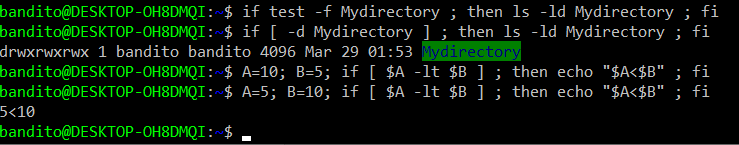
## Використання оператора if який використовує код повернення програми grep

Кожний процес Linux при завершенні передає батьківському **код повернення** (*exit status*), який дорівнює нулю, якщо процес вважає, що його робота була успішною, або *номеру помилки* у протилежному випадку. Що більше важливо, код повернення використовується в умовних операторах: якщо він дорівнює нулю,умова вважається виконаною, а якщо ні – не виконаною. **Умовний оператор *if***запускає команду-умову, ***grep -q***, яка нічого не виводить на екран, зате, повертає ***0***, якщо шаблон знайдено, і ***1***, якщо ні. В залежності від коду повернення цієї команди, ***if*** виконує або не виконує *тіло*: список команд, розміщених між ***then*** і ***fi***. Крапка з комою *розділяє* оператори в ***sh***;



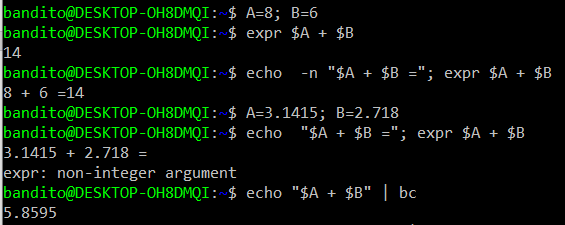
## Використання оператора test

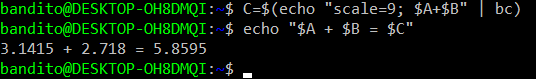
Багатьма функціями володіє команда ***test***: вона вміє порівнювати числа і рядки, перевіряти ярлик об’єкту файлової системи і наявність самого цього об’єкту. У «***test***» є друге ім’я: «[»(як правило, ***/usr/bin/[*** – символьне або навіть жорстке послання на ***/usr/bin/test***), яке дозволяє оформляти оператор ***if*** у більше звичному вигляді



## Підставлення виводу команди

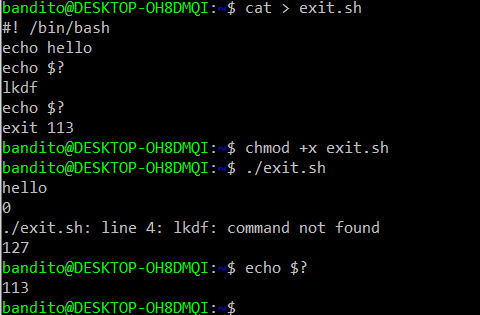
Другий тип підстановки, яку робить shell, робиться в середині подвійних лапок – це **підстановка виводу** команди. Підстановка виводу має вигляд «***’команда’***» (іншій варіант «***$(команда)***»). Як і підстановка значення змінної, вона відбувається *перед* тим, як почнеться розбір командного рядку: виконавши команду і отримавши від неї якійсь текст, shell береться розбирати його, так як би цей текст користувач набрав вручну. Це дуже зручний засіб, якщо те, що виводить команда, необхідно передати самому інтерпретатору





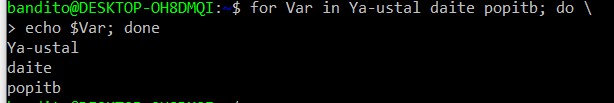
## Використання позиційних параметрів в сценарії

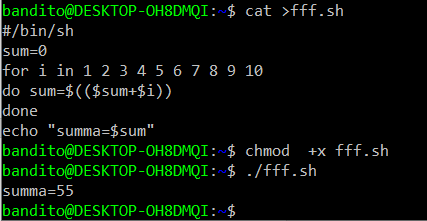
У мові **sh** багато уваги було приділено зручності написання **сценаріїв**. Зокрема, параметри командного рядку, які передані сценарію, доступні в ньому у вигляді змінних, імена яких співпадають з порядковим номером параметру



## Використання for і операції «.»

В ***sh*** є і оператор циклу ***while***, формат якого аналогічний ***if***, більше зручний саме у сценаріях оператор обходу списку ***for*** (список ділиться на слова також як і командний рядок – за допомогою **роздільників**)





Спочатку ***for*** виводить значення аргументів із введеного списку. Далі, ми пишемо файл-сценарій ***fff.sh***, в якому застосовуємо ***for*** для підрахунку суми перших десяти чисел натурального ряду. Потім запускаємо сценарій і отримуємо результат.

# **Висновок**

В ході виконання даної лабораторної роботи я визначив поняття і аналіз роботи нижче названих об’єктів та процесів, які відбуваються при доступі процесів до файлів і каталогів. Було здобуто знання щодо можливостей командної оболонки. Були опрацьовані команди для роботи з командною оболонкою.