

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

**“ЗАТВЕРДЖЕНО”**

Керівник роботи

Світлана ПОПЕРЕШНЯК

“20” листопада 2023 р.

**Файлова система з консольним інтерфейсом**

**Керівництво користувача**

КПІ.ІП-1123.045440.05.34

**“ПОГОДЖЕНО”**

Керівник роботи:

Світлана ПОПЕРЕШНЯК

Консультант:

Максим ГОЛОВЧЕНКО

Виконавець:

Сергій ПАНЧЕНКО

## Зміст

1 Призначення програми.....	3
2 Підготовка до роботи з програмним забезпеченням.....	4
2.1 Системні вимоги для коректної роботи.....	4
2.2 Завантаження застосунку.....	4
2.3 Перевірка коректної роботи.....	8
3 Виконання програми.....	10

## **1 ПРИЗНАЧЕННЯ ПРОГРАМИ**

СppFuse призначена для створення власної віртуальної файлової системи, яка може використовуватися для експериментів, тестування або власних розробок. Завдяки можливостям ВФС користувач може створити віртуальні директорії, файли, а також визначити особливості їхнього зчитування та запису.

## 2 ПІДГОТОВКА ДО РОБОТИ З ПРОГРАМНИМ ЗАБЕЗПЕЧЕННЯМ

### 2.1 Системні вимоги для коректної роботи

Для успішного функціонування програми користувач повинен мати IBM-сумісний персональний комп'ютер.

Мінімальна конфігурація технічних засобів:

- тип процесору: AMD® Ryzen 5 5500u with radeon graphics × 12;
- об'єм ОЗП: 4 Гб;
- тип відеокарти: RENOIR (renoir, LLVM 15.0.7, DRM 3.49, 6.2.0-33-generic);
- назва операційної системи: Ubuntu 20.04.3 LTS
- тип операційної системи: 64-бітна

Рекомендована конфігурація технічних засобів:

- тип процесору: AMD® Ryzen 5 5500u with radeon graphics × 12;
- об'єм ОЗП: 8 Гб;
- тип відеокарти: RENOIR (renoir, LLVM 15.0.7, DRM 3.49, 6.2.0-33-generic);
- назва операційної системи: Ubuntu 20.04.3 LTS
- тип операційної системи: 64-бітна

### 2.2 Завантаження застосунку

Для завантаження застосунку треба перейти до терміналу. Якщо ви користуєтеся Ubuntu Linux, то його можна відкрити декількома шляхами:

1. з допомогою комбінації клавіш CTRL + ALT + T;
2. з допомогою іконки на панелі задач, як показано на рисунку 2.1:



Рисунок 2.1 Іконка терміналу

3. з допомогою загального меню, до якого треба перейти за іконкою на рисунку 2.2 на панелі завдань, далі прописати у пошуковому рядку “terminal” на рисунку 2.3.



Рисунок 2.2 Іконка загального меню

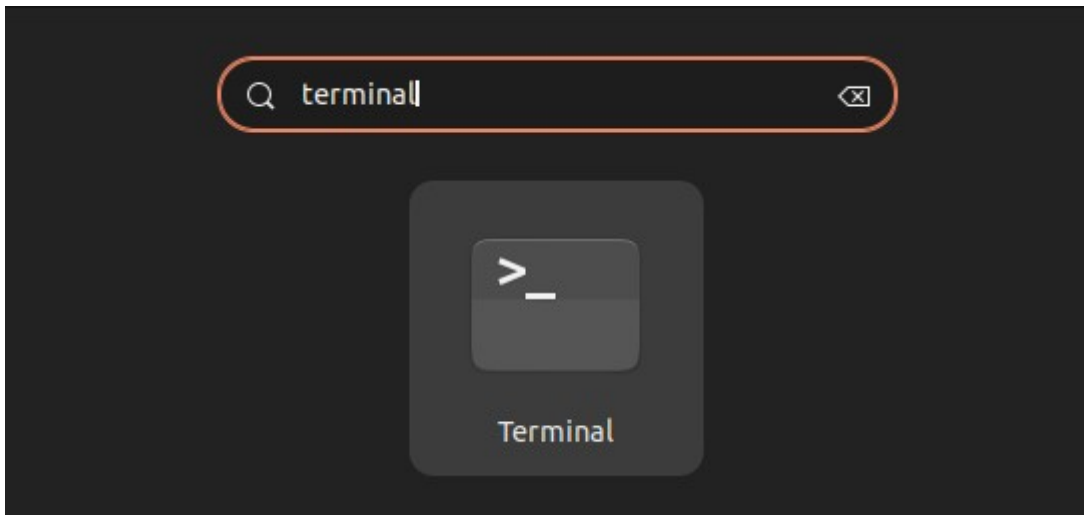


Рисунок 2.3 Уведення слова термінал у загальному пошуковому рядку

Після виконання однієї з вище зазначених дій ви маєте побачити перед собою відкритий термінал на рисунку 2.4.

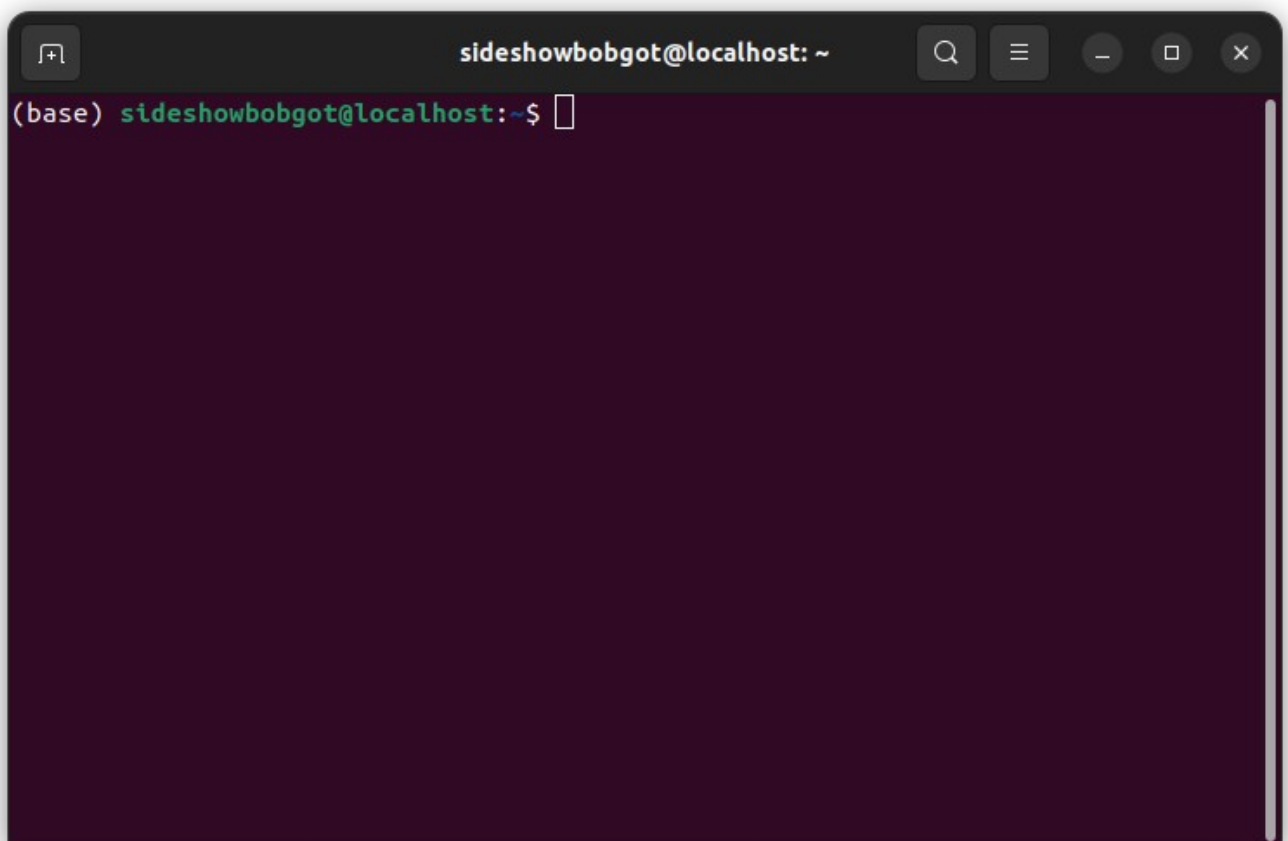
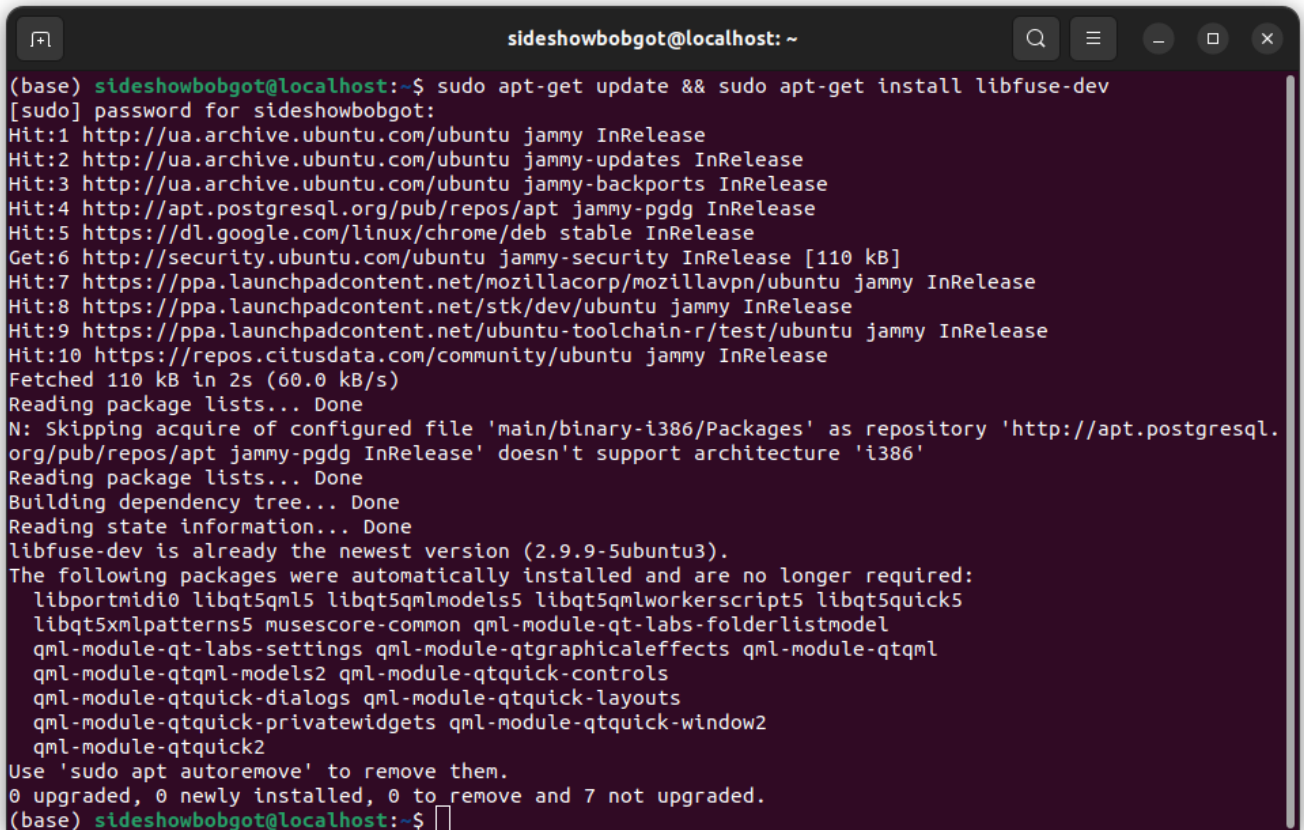


Рисунок 2.4 Відкритий термінал

Наступним кроком треба оновити пакети системи і для цього треба мати права адміністратора, тому використаємо команду “`sudo apt-get update -y && sudo apt-get install -y libfuse-dev`”. Вона оновить систему та встановить бібліотеку FUSE на

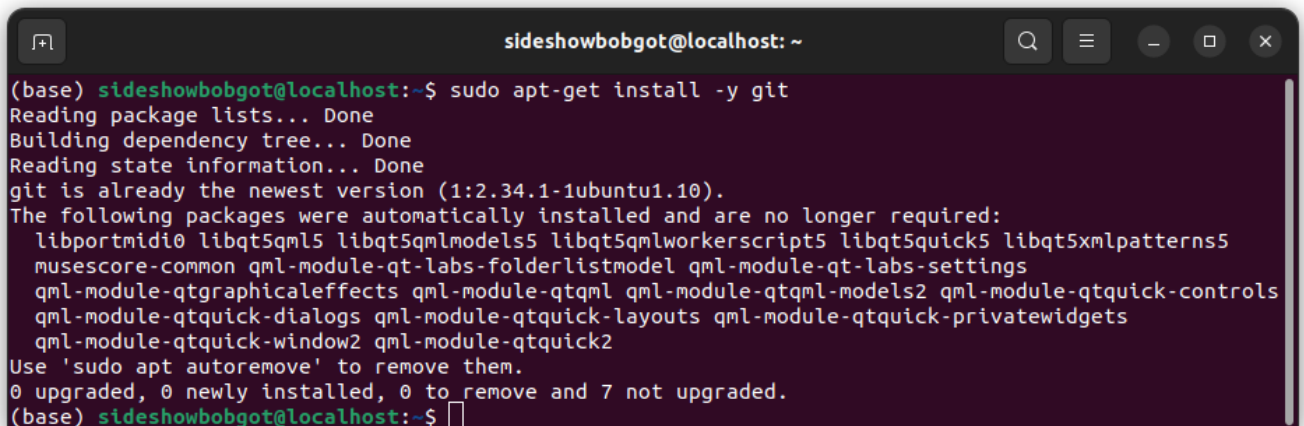
рисунку 2.5. Важливо зауважити, що у вас запитують пароль для вашого акаунта на ОС, при введенні пароля його не буде видно, тому уведіть його і натисніть Enter.



```
sideshowbobgot@localhost: ~  
(base) sideshowbobgot@localhost:~$ sudo apt-get update && sudo apt-get install libfuse-dev  
[sudo] password for sideshowbobgot:  
Hit:1 http://ua.archive.ubuntu.com/ubuntu jammy InRelease  
Hit:2 http://ua.archive.ubuntu.com/ubuntu jammy-updates InRelease  
Hit:3 http://ua.archive.ubuntu.com/ubuntu jammy-backports InRelease  
Hit:4 http://apt.postgresql.org/pub/repos/apt jammy-pgdg InRelease  
Hit:5 https://dl.google.com/linux/chrome/deb stable InRelease  
Get:6 http://security.ubuntu.com/ubuntu jammy-security InRelease [110 kB]  
Hit:7 https://ppa.launchpadcontent.net/mozillacorp/mozillavpn/ubuntu jammy InRelease  
Hit:8 https://ppa.launchpadcontent.net/stk/dev/ubuntu jammy InRelease  
Hit:9 https://ppa.launchpadcontent.net/ubuntu-toolchain-r/test/ubuntu jammy InRelease  
Hit:10 https://repos.citusdata.com/community/ubuntu jammy InRelease  
Fetched 110 kB in 2s (60.0 kB/s)  
Reading package lists... Done  
N: Skipping acquire of configured file 'main/binary-i386/Packages' as repository 'http://apt.postgresql.org/pub/repos/apt jammy-pgdg InRelease' doesn't support architecture 'i386'  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
libfuse-dev is already the newest version (2.9.9-5ubuntu3).  
The following packages were automatically installed and are no longer required:  
  libportmidi0 libqt5qml5 libqt5qmlmodels5 libqt5qmlworkerscript5 libqt5quick5  
  libqt5xmlpatterns5 musescore-common qml-module-qt-labs-folderlistmodel  
  qml-module-qt-labs-settings qml-module-qtgraphicaleffects qml-module-qtqml  
  qml-module-qtqml-models2 qml-module-qtquick-controls  
  qml-module-qtquick-dialogs qml-module-qtquick-layouts  
  qml-module-qtquick-privatewidgets qml-module-qtquick-window2  
  qml-module-qtquick2  
Use 'sudo apt autoremove' to remove them.  
0 upgraded, 0 newly installed, 0 to remove and 7 not upgraded.  
(base) sideshowbobgot@localhost:~$
```

Рисунок 2.5 Оновлення системи та завантаження бібліотеки FUSE

Далі треба переконатися, що у вас встановлена система контролю версій GIT. Прописуємо в терміналі “sudo apt-get install git” на рисунку 2.6.



```
sideshowbobgot@localhost: ~  
(base) sideshowbobgot@localhost:~$ sudo apt-get install -y git  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
git is already the newest version (1:2.34.1-1ubuntu1.10).  
The following packages were automatically installed and are no longer required:  
  libportmidi0 libqt5qml5 libqt5qmlmodels5 libqt5qmlworkerscript5 libqt5quick5 libqt5xmlpatterns5  
  musescore-common qml-module-qt-labs-folderlistmodel qml-module-qt-labs-settings  
  qml-module-qtgraphicaleffects qml-module-qtqml qml-module-qtqml-models2 qml-module-qtquick-controls  
  qml-module-qtquick-dialogs qml-module-qtquick-layouts qml-module-qtquick-privatewidgets  
  qml-module-qtquick-window2 qml-module-qtquick2  
Use 'sudo apt autoremove' to remove them.  
0 upgraded, 0 newly installed, 0 to remove and 7 not upgraded.  
(base) sideshowbobgot@localhost:~$
```

Рисунок 2.6 Встановлення GIT

Склонуйте репозиторій CppFuse з GitHub “git clone

<https://github.com/SideShowBoBGOT/CppFuse>” на рисунку 2.7.

```
sideshowbobgot@localhost: ~  
(base) sideshowbobgot@localhost:~$ git clone https://github.com/SideShowBoBGOT/CppFuse  
Cloning into 'CppFuse'...  
remote: Enumerating objects: 1184, done.  
remote: Counting objects: 100% (122/122), done.  
remote: Compressing objects: 100% (89/89), done.  
remote: Total 1184 (delta 47), reused 88 (delta 31), pack-reused 1062  
Receiving objects: 100% (1184/1184), 17.84 MiB | 1.10 MiB/s, done.  
Resolving deltas: 100% (689/689), done.  
(base) sideshowbobgot@localhost:~$
```

Рисунок 2.7 Клонування репозиторія CppFuse

Далі потрібно встановити усі залежні бібліотеки репозиторія. Перейдемо у директорію репозиторія “cd CppFuse” та пропишемо команду “git submodule update --init --recursive” на рисунку 2.8.

```
sideshowbobgot@localhost: ~/CppFuse  
(base) sideshowbobgot@localhost:~$ cd CppFuse/  
(base) sideshowbobgot@localhost:~/CppFuse$ git submodule update --init --recursive  
Submodule 'External/CLI11' (https://github.com/CLIUtils/CLI11) registered for path 'External/CLI11'  
Submodule 'External/RwLock' (https://github.com/SideShowBoBGOT/RwLock) registered for path 'External/RwLock'  
Submodule 'External/googletest' (https://github.com/google/googletest) registered for path 'External/googletest'  
Submodule 'External/magic_enum' (https://github.com/Neargye/magic_enum) registered for path 'External/magic_enum'  
Cloning into '/home/sideshowbobgot/CppFuse/External/CLI11'...  
Cloning into '/home/sideshowbobgot/CppFuse/External/RwLock'...  
Cloning into '/home/sideshowbobgot/CppFuse/External/googletest'...  
Cloning into '/home/sideshowbobgot/CppFuse/External/magic_enum'...  
Submodule path 'External/CLI11': checked out 'b8fc255e97d215c14c611ac352a9668ecdd425ee'  
Submodule path 'External/RwLock': checked out '7bf1989b4911df63373e089e56f389939ef18a96'  
Submodule path 'External/googletest': checked out 'b10fad38c4026a29ea6561ab15fc4818170d1c10'  
Submodule path 'External/magic_enum': checked out '3437129f30c926e740ae5943cef27cd0b836c4be'  
(base) sideshowbobgot@localhost:~/CppFuse$
```

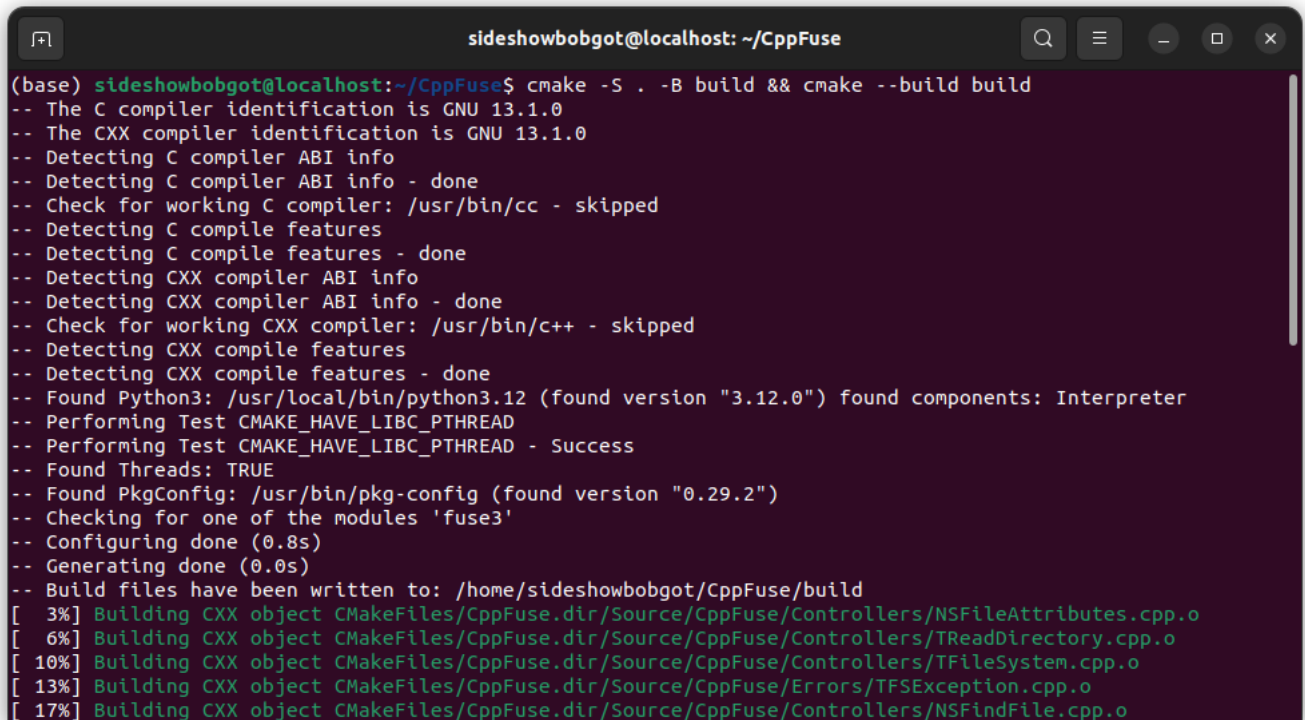
Рисунок 2.8 Встановлення залежних бібліотек для репозиторія

Далі встановимо систему білдигу проєктів Смаке для цього у командному рядку прописуємо команду “sudo apt-get install cmake” на рисунку 2.9.

```
sideshowbobgot@localhost: ~  
(base) sideshowbobgot@localhost:~$ sudo apt-get install -y cmake  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
cmake is already the newest version (3.22.1-1ubuntu1.22.04.1).  
The following packages were automatically installed and are no longer required:  
  libportmidi0 libqt5qml5 libqt5qmlmodels5 libqt5qmlworkerscript5 libqt5quick5  
  libqt5xmlpatterns5 musescore-common qml-module-qt-labs-folderlistmodel  
  qml-module-qt-labs-settings qml-module-qtgraphicaleffects qml-module-qtqml  
  qml-module-qtqml-models2 qml-module-qtquick-controls qml-module-qtquick-dialogs  
  qml-module-qtquick-layouts qml-module-qtquick-privatewidgets  
  qml-module-qtquick-window2 qml-module-qtquick2  
Use 'sudo apt autoremove' to remove them.  
0 upgraded, 0 newly installed, 0 to remove and 7 not upgraded.  
(base) sideshowbobgot@localhost:~$
```

Рисунок 2.9 Встановлення Смаке

Перейдіть до створеного каталогу репозиторія “cd CppFuse” та збудимо застосунок з допомогою команди “cmake -S . -B build && cmake --build build” на рисунку 2.10.

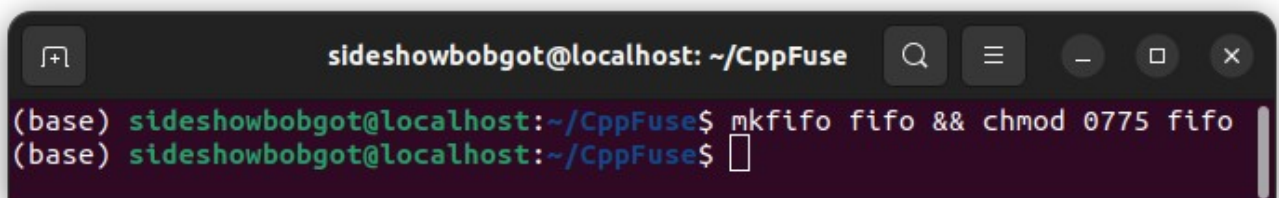


```
sideshowbobgot@localhost: ~/CppFuse
(base) sideshowbobgot@localhost:~/CppFuse$ cmake -S . -B build && cmake --build build
-- The C compiler identification is GNU 13.1.0
-- The CXX compiler identification is GNU 13.1.0
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Check for working C compiler: /usr/bin/cc - skipped
-- Detecting C compile features
-- Detecting C compile features - done
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Check for working CXX compiler: /usr/bin/c++ - skipped
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Found Python3: /usr/local/bin/python3.12 (found version "3.12.0") found components: Interpreter
-- Performing Test CMAKE_HAVE_LIBC_PTHREAD
-- Performing Test CMAKE_HAVE_LIBC_PTHREAD - Success
-- Found Threads: TRUE
-- Found PkgConfig: /usr/bin/pkg-config (found version "0.29.2")
-- Checking for one of the modules 'fuse3'
-- Configuring done (0.8s)
-- Generating done (0.0s)
-- Build files have been written to: /home/sideshowbobgot/CppFuse/build
[ 3%] Building CXX object CMakeFiles/CppFuse.dir/Source/CppFuse/Controllers/NSFileAttributes.cpp.o
[ 6%] Building CXX object CMakeFiles/CppFuse.dir/Source/CppFuse/Controllers/TReadDirectory.cpp.o
[ 10%] Building CXX object CMakeFiles/CppFuse.dir/Source/CppFuse/Controllers/TFileSystem.cpp.o
[ 13%] Building CXX object CMakeFiles/CppFuse.dir/Source/CppFuse/Errors/TFSEException.cpp.o
[ 17%] Building CXX object CMakeFiles/CppFuse.dir/Source/CppFuse/Controllers/NSFindFile.cpp.o
```

Рисунок 2.10 Білдинг застосунку

## 2.3 Перевірка коректної роботи

Для розгортання ВФС потрібно створити комунікаційний файл через, який команда швидкого пошуку CppFuse буде звертатися до ВФС. У каталозі CppFuse пропишемо команду “mkfifo fifo && chmod 0775 fifo”, щоб створити цей комунікаційний файл та надати йому усі необхідні права виконання на рисунку 2.11.

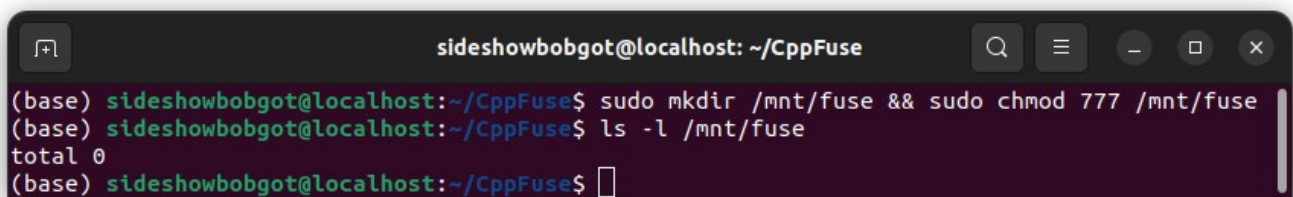


```
sideshowbobgot@localhost: ~/CppFuse
(base) sideshowbobgot@localhost:~/CppFuse$ mkfifo fifo && chmod 0775 fifo
(base) sideshowbobgot@localhost:~/CppFuse$
```

Рисунок 2.11 Створення комінкаційного фала та надання йому прав

Далі створимо директорію для ВФС у директорії монтування /mnt, куди зазвичай під’єднуються зовнішні носії даних на рисунку 2.12. Пропишемо команду “mkdir /mnt/fuse && sudo chmod 0777 /mnt/fuse”.

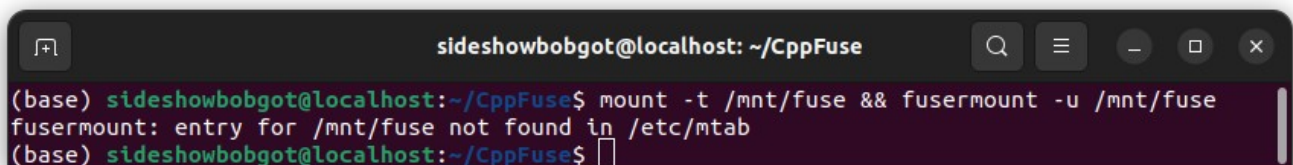


A terminal window titled 'sideshowbobgot@localhost: ~/CppFuse' with search, menu, and window control icons. The terminal shows the following commands and output:

```
(base) sideshowbobgot@localhost:~/CppFuse$ sudo mkdir /mnt/fuse && sudo chmod 777 /mnt/fuse
(base) sideshowbobgot@localhost:~/CppFuse$ ls -l /mnt/fuse
total 0
(base) sideshowbobgot@localhost:~/CppFuse$
```

Рисунок 2.12 Створення директорії для ВФС

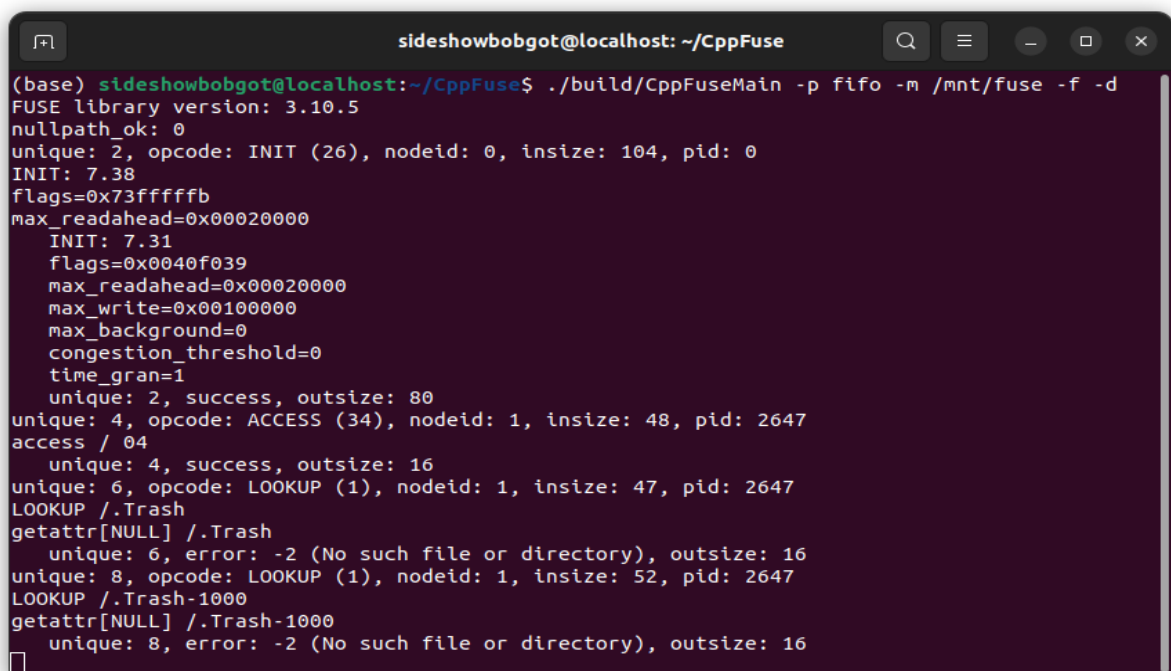
Далі приєднаємо ВФС до даної директорії “mount -t /mnt/fuse && fusermount -u /mnt/fuse” на рисунку 2.13.

A terminal window titled 'sideshowbobgot@localhost: ~/CppFuse' with search, menu, and window control icons. The terminal shows the following commands and output:

```
(base) sideshowbobgot@localhost:~/CppFuse$ mount -t /mnt/fuse && fusermount -u /mnt/fuse
fusermount: entry for /mnt/fuse not found in /etc/mtab
(base) sideshowbobgot@localhost:~/CppFuse$
```

Рисунок 2.13 Приєднання ВФС

Далі для перевірки коректності роботи запустимо ВФС. Запустимо ВФС з допомогою команди “./build/CppFuseMain -p fifo -m /mnt/fuse -f -d”. Маємо спостерігати ініціалізацію ВФС.

A terminal window titled 'sideshowbobgot@localhost: ~/CppFuse' with search, menu, and window control icons. The terminal shows the following commands and output:

```
(base) sideshowbobgot@localhost:~/CppFuse$ ./build/CppFuseMain -p fifo -m /mnt/fuse -f -d
FUSE library version: 3.10.5
nullpath_ok: 0
unique: 2, opcode: INIT (26), nodeid: 0, insize: 104, pid: 0
INIT: 7.38
flags=0x73fffffb
max_readahead=0x00020000
  INIT: 7.31
  flags=0x0040f039
  max_readahead=0x00020000
  max_write=0x00100000
  max_background=0
  congestion_threshold=0
  time_gran=1
unique: 2, success, outsize: 80
unique: 4, opcode: ACCESS (34), nodeid: 1, insize: 48, pid: 2647
access / 04
  unique: 4, success, outsize: 16
unique: 6, opcode: LOOKUP (1), nodeid: 1, insize: 47, pid: 2647
LOOKUP /.Trash
getattr[NULL] /.Trash
  unique: 6, error: -2 (No such file or directory), outsize: 16
unique: 8, opcode: LOOKUP (1), nodeid: 1, insize: 52, pid: 2647
LOOKUP /.Trash-1000
getattr[NULL] /.Trash-1000
  unique: 8, error: -2 (No such file or directory), outsize: 16

```

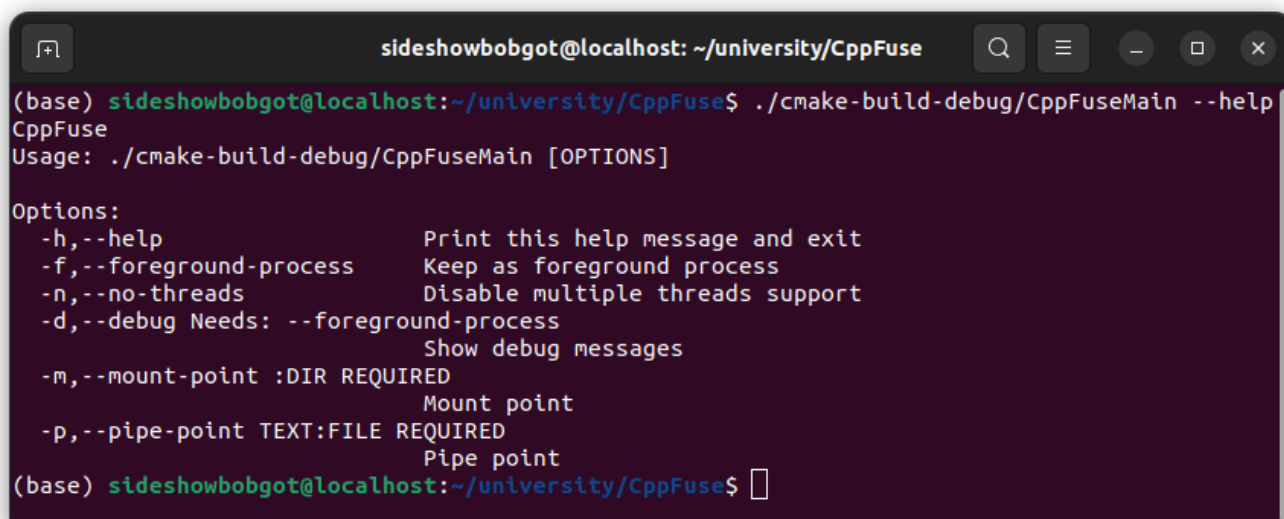
Рисунок 2.14 Перевірка запуску ВФС

### 3 ВИКОНАННЯ ПРОГРАМИ

Для перегляду усіх команд використовуйте прапорець `--help` на рисунку 3.1.

Використовуйте команди та аргументи командного рядка для налаштування роботи ВФС згідно власних потреб:

- `-f, --foreground-process`: Необов'язковий аргумент, який утримує ВФС як фронтальний процес, тобто утримує на передньому плані.
- `-n, --no-threads`: Необов'язковий аргумент для вимкнення підтримки багатопотоковості, що може бути корисним у випадках, коли бажано вимкнути опцію використання багатопотоковості.
- `-d, --debug`: Обов'язковий разом із `-f`. Включає відображення символів налагодження для отримання відладкової інформації.
- `-m, --mount-point`: Обов'язковий аргумент, що визначає точку приєднання ВФС до основної файлової системи.
- `-p, --pipe-point`: Обов'язковий аргумент, що визначає точку FIFO-файла для встановлення зв'язку та комунікації з ВФС.



```
sideshowbobgot@localhost: ~/university/CppFuse
(base) sideshowbobgot@localhost:~/university/CppFuse$ ./cmake-build-debug/CppFuseMain --help
CppFuse
Usage: ./cmake-build-debug/CppFuseMain [OPTIONS]

Options:
  -h,--help                Print this help message and exit
  -f,--foreground-process   Keep as foreground process
  -n,--no-threads           Disable multiple threads support
  -d,--debug Needs: --foreground-process
                           Show debug messages
  -m,--mount-point :DIR REQUIRED
                           Mount point
  -p,--pipe-point TEXT:FILE REQUIRED
                           Pipe point
(base) sideshowbobgot@localhost:~/university/CppFuse$
```

Рисунок 3.1 Виведення основних прапорців виконання ВФС

Для використання команди швидкого пошуку перейдіть у директорі `Scripts` всередині `CppFuse` “`cd Scripts`” та відкрийте файл “`nano CppFuseFind`”. Відредагуйте повний шлях до комунікаційного файлу та бінарного файлу `CppFuseFind` на рисунку 3.2.

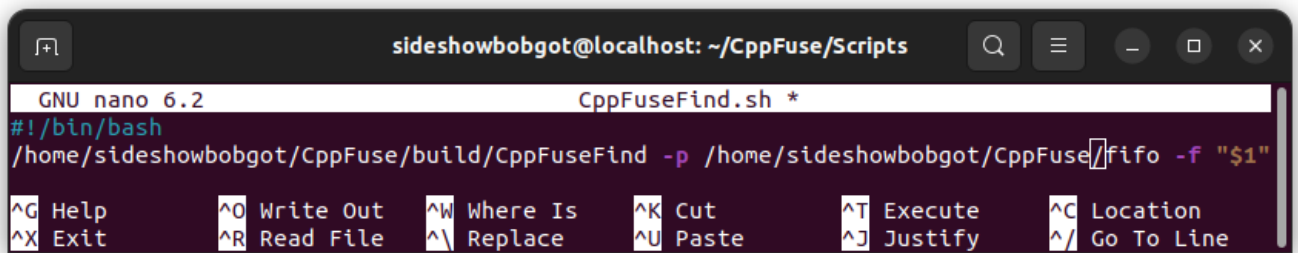


Рисунок 3.2 Редагування скрипта CppFuseFind.sh

Далі скопіюйте цей скрипт до каталогу бінарних файлів системи, а саме: /usr/bin на рисунку 3.3.

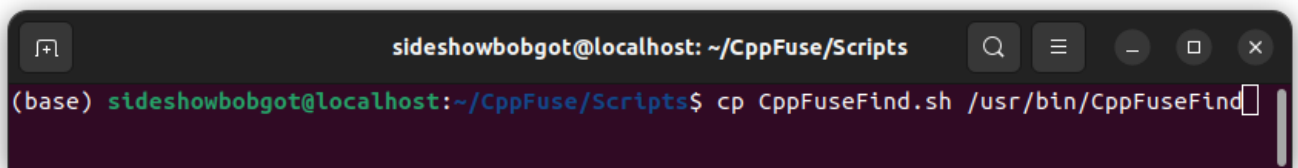


Рисунок 3.3 Копіювання скрипта до /usr/bin

На рисунку 3.4 можна побачити основні прапорці виконання CppFuseFind:

1. -h, --help – команда виклику допомоги;
2. -p, --pipe-point - обов'язковий аргумент, що визначає точку FIFO-файла для встановлення зв'язку та комунікації з ВФС;
3. -f, --file-name – ім'я файлу, за яким буде виконуватися пошук.

Використовуйте скрипт CppFuseFind, передавши у нього назву відповідного файлу, що можна побачити на рисунку 3.5.

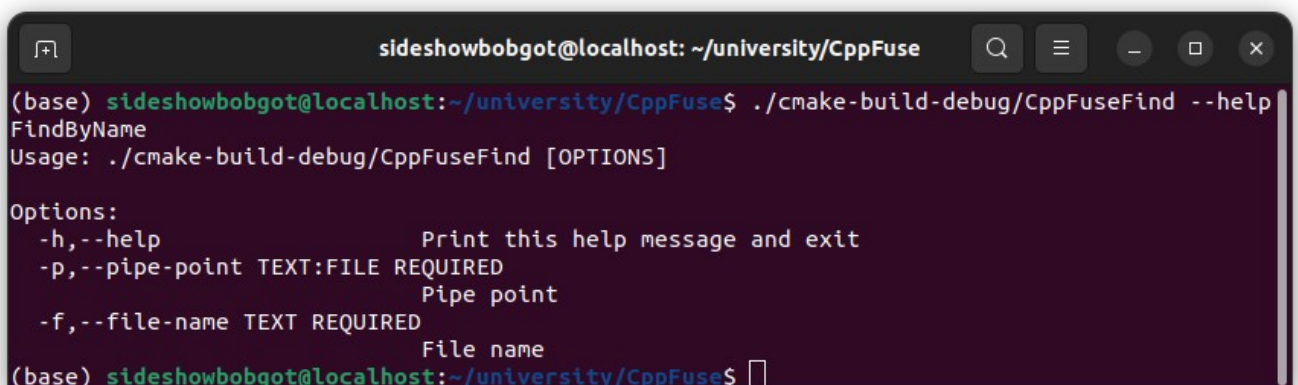


Рисунок 3.4 Основні прапорці виконання CppFuseFind

```
sideshowbobgot@localhost: /mnt...
sideshowbobgot@localhost:/mnt/fuse$ mkdir a b c d e
sideshowbobgot@localhost:/mnt/fuse$ touch a/text
touch: setting times of 'a/text': Function not implemented
sideshowbobgot@localhost:/mnt/fuse$ touch b/text
touch: setting times of 'b/text': Function not implemented
sideshowbobgot@localhost:/mnt/fuse$ touch c/text
touch: setting times of 'c/text': Function not implemented
sideshowbobgot@localhost:/mnt/fuse$ touch d/text
touch: setting times of 'd/text': Function not implemented
sideshowbobgot@localhost:/mnt/fuse$ touch e/text
touch: setting times of 'e/text': Function not implemented
sideshowbobgot@localhost:/mnt/fuse$ CppFuseFind text
--file-name is required
Run with --help for more information.
sideshowbobgot@localhost:/mnt/fuse$ CppFuseFind -f text
--file-name: 1 required TEXT missing
Run with --help for more information.
sideshowbobgot@localhost:/mnt/fuse$ CppFuseFind text
/a/text
/b/text
/c/text
/d/text
/dir/text
/e/text
sideshowbobgot@localhost:/mnt/fuse$
```

Рисунок 3.5 Знаходження файла з допомогою CppFuseFind