Завдання 4

Оскільки завдання 4 включає в себе завдані 1-3, то почнемо роботу з завдання 4.

Програма працює як простий командний процесор, підтримуючи стандартні команди оболонки через system() і дозволяє налаштовувати розмір буфера для введення команд.

```
mint@asus-mint:~/git/op-course/lab3/task4$ ./bin/main -h
Usage: ./simple_shell [OPTIONS]
Options:
  -h, --help           Show this help message
  -b, --buffer SIZE    Set buffer size for command input (default: 127)
mint@asus-mint:~/git/op-course/lab3/task4$ ./bin/main -b 256
Welcome to the simple shell processor!
User: mint
[mint]$ ls
bin  build  Makefile  src
[mint]$ fail
sh: 1: fail: not found
[mint]$ stop

Wait 3 seconds...
```

Рисунок 1 – Результат виконання завдання 4.

Завдання 5

Програма приймає кілька опцій командного рядка. Якщо вказано опцію -f або --file, змінна середовища FILE_NAME отримує вказане значення. Опція -n або --number визначає кількість дочірніх процесів, які будуть створені програмою. Кожен дочірній процес створюється за допомогою функції fork() і виконує окрему програму. Ця програма перевіряє наявність змінної середовища FILE_NAME. Якщо змінна визначена, базове ім'я файлу буде взято з її значення, і до нього додається порядковий номер дочірнього процесу. Якщо ж змінна не задана, програма використовує ім'я за замовчуванням.

```
mint@asus-mint:~/git/op-course/lab3/task5$ ./bin/main
Environment variable FILE_NAME not set
Child process 1: generating random numbers into file output_file_1.txt
mint@asus-mint:~/git/op-course/lab3/task5$ ./bin/main --file myfile --number 3
Environment variable FILE_NAME set to myfile
Child process 1: generating random numbers into file myfile_1.txt
Child process 2: generating random numbers into file myfile_2.txt
Child process 3: generating random numbers into file myfile_3.txt
mint@asus-mint:~/git/op-course/lab3/task5$ cat myfile_1.txt
0.840188
```

Рисуно 2 – Результат виконання завдання 5.

Висновок

Під час виконання цієї роботи було успішно реалізовано кілька програм, що демонструють основи роботи з системними викликами, середовищем виконання та процесами в операційній системі Linux. Зокрема, було розглянуто наступні важливі аспекти:

1. Робота з системними змінними середовища: Одна з реалізованих програм показала, як можна встановлювати, змінювати та видаляти змінні середовища, що є основним елементом взаємодії між процесами та системою. Була реалізована програма, яка за допомогою різних опцій надає доступ до системних змінних, що дозволило детальніше зрозуміти їхню роль у програмуванні в Linux.
2. Робота з процесами: Було продемонстровано використання системних викликів для створення дочірніх процесів за допомогою функції fork() та очікування їх завершення за допомогою wait(). Ці механізми є основними для управління багатозадачністю на рівні операційної системи.
3. Передача аргументів між процесами: Була реалізована передача даних у вигляді системних змінних або аргументів командного рядка між батьківськими та дочірніми процесами. Це дозволило дочірнім процесам працювати з різними файлами, базуючись на отриманих аргументах.
4. Робота з файлами та введенням/виведенням: Програми, що генерують випадкові числа та записують їх у файли, продемонстрували ефективне використання файлової системи, зокрема відкриття, запис та обробка помилок при роботі з файлами.
5. Структурованість проєкту: Важливим етапом було розбиття проєктів на багатофайлові структури, що сприяло кращій організації коду та полегшенню його підтримки. Було створено Makefile для автоматизації компіляції, що забезпечує гнучкість при роботі з різними типами бібліотек та залежностями.