

**РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ
НАРОДОВ**

**Факультет физико-математических и естественных
наук**

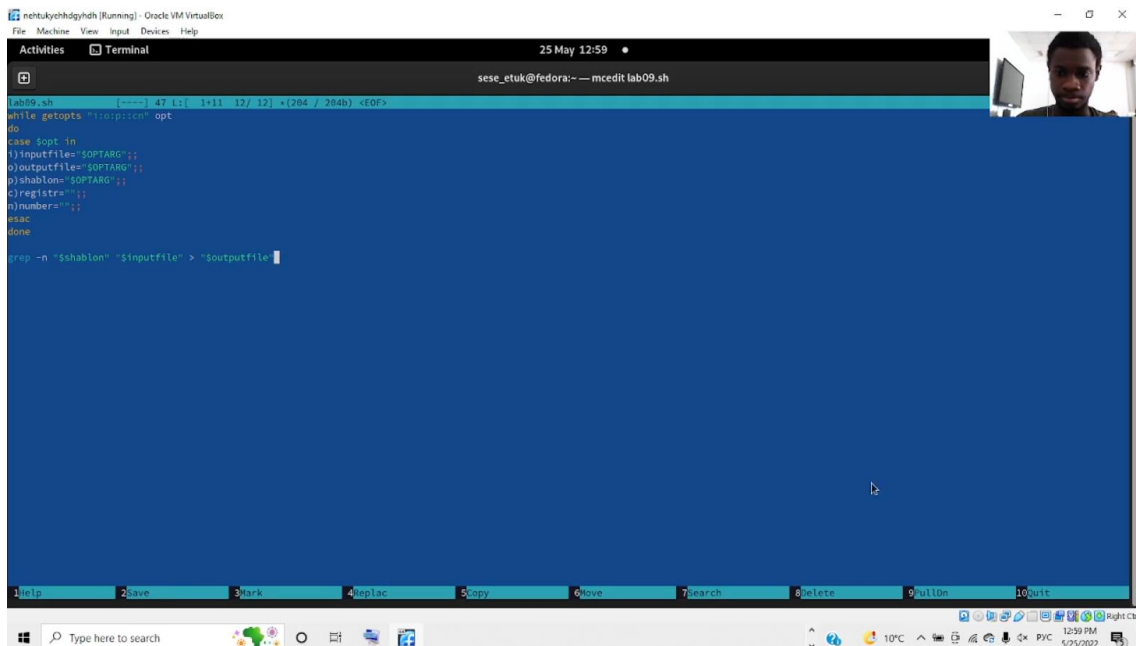
**Фундаментальная Информатика и Информационные
технологии**

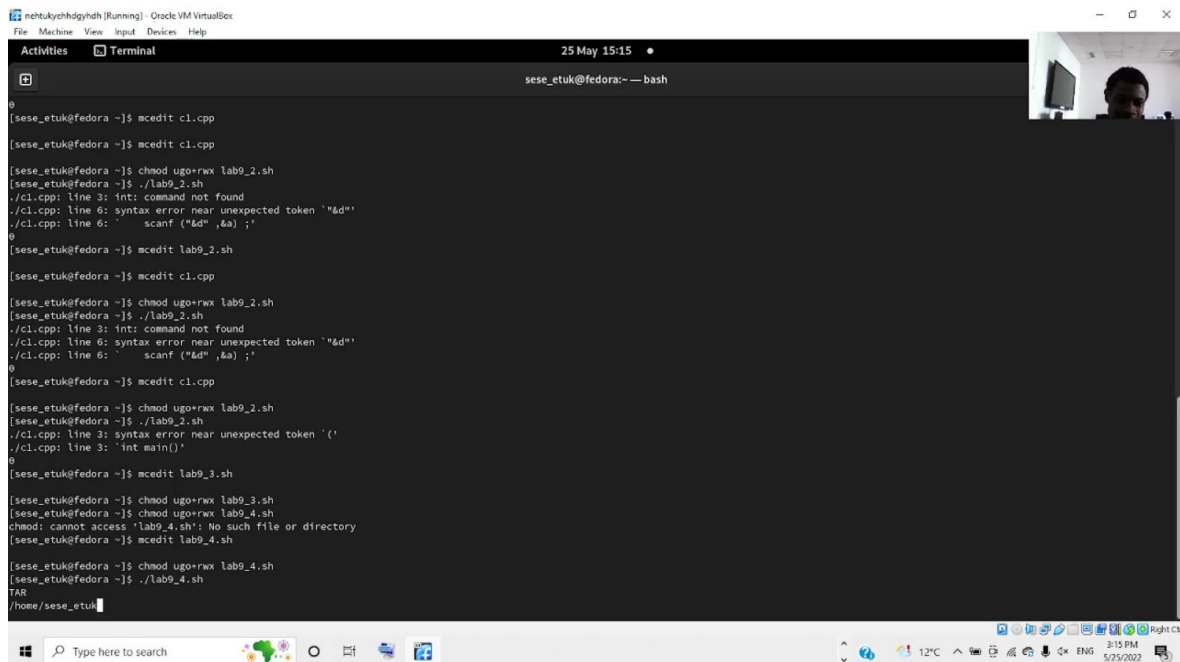
ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 11

дисциплина: операционные системы

Этук Нсе-Абаси Акпан

НФИбд-02-21





```
nehtukyhdyhdyh [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
25 May 15:15
sese_etuk@fedora:~$ bash

[sese_etuk@fedora ~]$ mcedit c1.cpp
[sese_etuk@fedora ~]$ mcedit c1.cpp

[sese_etuk@fedora ~]$ chmod ugo+rx lab9_2.sh
[sese_etuk@fedora ~]$ ./lab9_2.sh
./c1.cpp: line 3: int: command not found
./c1.cpp: line 6: syntax error near unexpected token '&d'
./c1.cpp: line 6: `scanf("&d", &a);'
[sese_etuk@fedora ~]$ mcedit lab9_2.sh
[sese_etuk@fedora ~]$ mcedit c1.cpp

[sese_etuk@fedora ~]$ chmod ugo+rx lab9_2.sh
[sese_etuk@fedora ~]$ ./lab9_2.sh
./c1.cpp: line 3: int: command not found
./c1.cpp: line 6: syntax error near unexpected token '&d'
./c1.cpp: line 6: `scanf("&d", &a);'
[sese_etuk@fedora ~]$ mcedit c1.cpp
[sese_etuk@fedora ~]$ chmod ugo+rx lab9_2.sh
[sese_etuk@fedora ~]$ ./lab9_2.sh
./c1.cpp: line 3: syntax error near unexpected token '('
./c1.cpp: line 3: `int main()'
[sese_etuk@fedora ~]$ mcedit lab9_3.sh

[sese_etuk@fedora ~]$ chmod ugo+rx lab9_3.sh
[sese_etuk@fedora ~]$ chmod ugo+rx lab9_4.sh
chmod: cannot access 'lab9_4.sh': No such file or directory
[sese_etuk@fedora ~]$ mcedit lab9_4.sh

[sese_etuk@fedora ~]$ chmod ugo+rx lab9_4.sh
[sese_etuk@fedora ~]$ ./lab9_4.sh
TAR
/home/sese_etuk
```

Вывод

Я изучил основы программирования в оболочке ОС UNIX, научился писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

Контрольные вопросы

1. Весьма необходимой при программировании является команда `getopts`, которая осуществляет синтаксический анализ командной строки, выделяя флаги, и используется для объявления переменных. Синтаксис команды следующий: `getopts option-string variable [arg ...]` Флаги – это опции командной строки, обычно помеченные знаком минус; Например, `-F` является флагом для команды `ls -F`. Иногда эти флаги имеют аргументы, связанные с ними. Программы интерпретируют эти флаги, соответствующим образом изменяя свое поведение. Строка опций `optionstring` — это список возможных букв и чисел соответствующего флага. Если ожидается, что некоторый флаг будет сопровождаться некоторым аргументом, то за этой буквой должно следовать двоеточие. Соответствующей переменной присваивается буква данной опции. Если команда `getopts` может распознать аргумент, она возвращает истину. Принято включать `getopts` в цикл `while` и анализировать введенные данные с помощью оператора `case`. Предположим, необходимо распознать командную строку следующего формата: `testprog -ifile_in.txt -ofile_out.doc -L -t -r` Вот как выглядит использование оператора `getopts` в этом случае: `while getopts o:i:Ltr optletter do case $optletter in o) oflag=1; oval=$OPTARG;; i) iflag=1; ival=$OPTARG;; L) Lflag=1;; t) tflag=1;; r) rflag=1;; *) echo Illegal option $optletter esac done` Функция `getopts` включает две специальные переменные среды – `OPTARG` и `OPTIND`. Если ожидается

дополнительное значение, то OPTARG устанавливается в значение этого аргумента (будет равна file_in.txt для опции i и file_out.doc для опции o). OPTIND является числовым индексом на упомянутый аргумент. Функция getopt также понимает переменные типа массив, следовательно, можно использовать ее в функции не только для синтаксического анализа аргументов функций, но и для анализа введенных пользователем данных.

2. При перечислении имен файлов текущего каталога можно использовать следующие символы: `*` — соответствует произвольной, в том числе и пустой строке; `?` — соответствует любому одному символу; `[c1-c2]` — соответствует любому символу, лексикографически находящемуся между символами c1 и c2. `echo *` — выведет имена всех файлов текущего каталога, что представляет собой простейший аналог команды `ls`; `ls .c` — выведет все файлы с последними двумя символами, равными `.c`. `echo prog.?` — выдаст все файлы, состоящие из пяти или шести символов, первыми пятью символами которых являются `prog.` `[a-z]` — соответствует произвольному имени файла в текущем каталоге, начинающемуся с любой строчной буквы латинского алфавита.

3. Часто бывает необходимо обеспечить проведение каких-либо действий циклически и управление дальнейшими действиями в зависимости от результатов проверки некоторого условия. Для решения подобных задач язык программирования `bash` предоставляет Вам возможность использовать такие управляющие конструкции, как `for`, `case`, `if` и `while`. С точки зрения командного процессора эти управляющие конструкции являются обычными командами и могут использоваться как при создании командных файлов, так и при работе в интерактивном режиме. Команды, реализующие подобные конструкции, по сути дела являются операторами языка программирования `bash`. Поэтому при описании языка программирования `bash` термин оператор будет использоваться наравне с термином команда.

4. Два несложных способа позволяют вам прерывать циклы в оболочке `bash`. Команда `break` завершает выполнение цикла, а команда `continue` завершает данную итерацию блока операторов. Команда `break` полезна для завершения цикла `while` в ситуациях, когда условие перестает быть правильным. Пример бесконечного цикла `while`, с прерыванием в момент, когда файл перестает существовать: `while true do if [! -f $file] then break fi sleep 10 done`

5. Команды ОС UNIX возвращают код завершения, значение которого может быть использовано для принятия решения о дальнейших действиях. Команда `test`, например, создана специально для использования в командных файлах. Единственная функция этой команды заключается в выработке кода завершения.

6. Введенная строка означает условие существования файла `man$s/$i.$s`

7. Если речь идет о 2-х параллельных действиях, то это `while`. когда мы показываем, что сначала делается 1-е действие. потом оно заканчивается при наступлении 2-го действия, применяем `until`.