

Отчёт по лабораторной работе №11

Операционные системы

Луангсуваннавонг Сайпхачан

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	9
4.1	Основные команды emacs	9
4.2	Управление буферами	15
4.3	Управление окнами	16
4.4	Режим поиска	18
5	Выводы	22
6	Ответы на контрольные вопросы	23
	Список литературы	25

Список иллюстраций

4.1	Окно Emacs	9
4.2	Создание нового файла	9
4.3	Добавление текстового кода	10
4.4	Сохранение файла	10
4.5	Вырезание целой линии	11
4.6	Вставка строки	11
4.7	Выделение текста и его копирование	12
4.8	Вставка строки	12
4.9	Вырезание целой линии	13
4.10	Отмена последнего действия	13
4.11	Текстовый код	13
4.12	Курсор в начале строки	13
4.13	Курсор в конце строки	14
4.14	Курсор в начале файла	14
4.15	Курсор в конце файла	15
4.16	Список активных буферов	15
4.17	Заккрытие буфера	16
4.18	Переключение на буфер	16
4.19	Буфер Buffer List	16
4.20	Разделение буферного кадра	17
4.21	Создание новых буферов	17
4.22	Режим поиска	18
4.23	Режим поиска	18
4.24	Переключение между результатами	19
4.25	Выход из режима поиска	19
4.26	Режим поиска и замены	19
4.27	Замена слова	20
4.28	Другой режим поиска	21

Список таблиц

1 Цель работы

Познакомиться с операционной системой Linux. Получить практические навыки работы с редактором Emacs.

2 Задание

1. Ознакомиться с теоретическим материалом.
2. Ознакомиться с редактором etas.
3. Выполнить упражнения.
4. Ответить на контрольные вопросы.

3 Теоретическое введение

Emacs (Editor MACroS) — один из самых гибких и настраиваемых текстовых редакторов, изначально разработанный Ричардом Столлманом в 1970-х годах в рамках проекта GNU. Он быстро вышел за рамки обычного редактора, превратившись в полноценную рабочую среду, включающую инструменты для программирования, написания документов, чтения почты, управления файлами и даже игр.

Основу архитектуры Emacs составляет язык Emacs Lisp — диалект Lisp, который позволяет пользователям настраивать, автоматизировать и расширять практически любой аспект редактора. Большая часть интерфейса и функций написана именно на Elisp, тогда как базовая система реализована на C.

Ключевые особенности:

Буферно-ориентированная работа — данные обрабатываются в буферах, которые могут содержать файлы, результаты команд, терминалы и другое.

Режимы (modes) — специальные режимы под разные типы задач: от org-mode для заметок до python-mode для программирования.

Интегрированная среда — поддержка компиляции, контроля версий (например, через Magit), терминала и отладки в рамках одного окна.

Несмотря на высокую кривую обучения (необычные сочетания клавиш и необходимость изучения Elisp), Emacs остаётся любимым инструментом разработчиков и продвинутых пользователей благодаря своей гибкости, расширяемости и философии «редактор, подстраивающийся под пользователя». Он объединяет принципы UNIX-инструментария в рамках одной мощной, программируемой

среды.

4 Выполнение лабораторной работы

4.1 Основные команды emacs

Я устанавливаю и запускаю emacs через терминал (рис. 4.1)

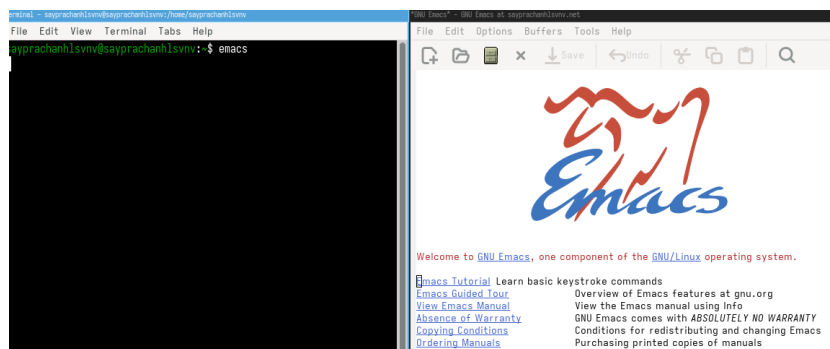


Рис. 4.1: Окно Emacs

Используя комбинацию клавиш Ctrl-x Ctrl-f (C-x C-f), я создаю файл lab07.sh (Поскольку он не находит файл, создается новый) (рис. 4.2)

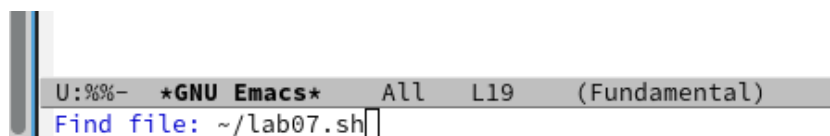
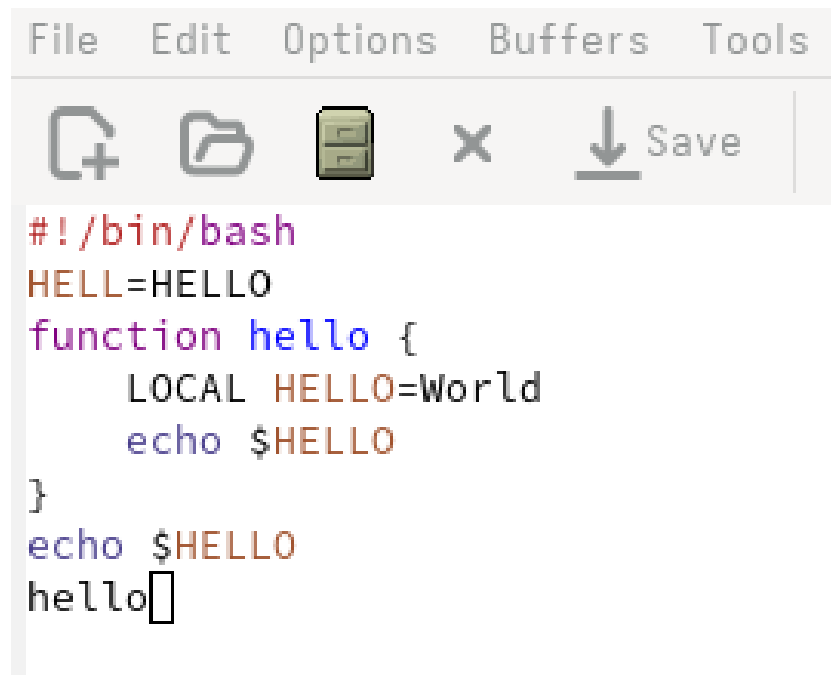


Рис. 4.2: Создание нового файла

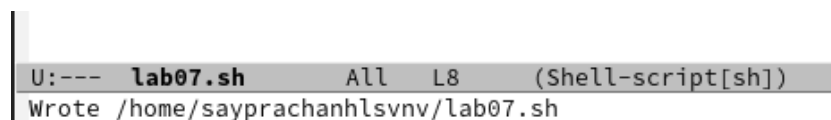
Добавляю текстовый код в файл lab07.sh (рис. 4.3)

A screenshot of a text editor window. The menu bar at the top includes 'File', 'Edit', 'Options', 'Buffers', and 'Tools'. Below the menu bar is a toolbar with icons for creating a new file, opening a file, saving a file, closing a file, and a 'Save' button with a downward arrow icon. The main text area contains a shell script with the following content:

```
#!/bin/bash
HELL=HELLO
function hello {
    LOCAL HELLO=World
    echo $HELLO
}
echo $HELLO
hello
```

Рис. 4.3: Добавление текстового кода

Сохраняю файл lab07.sh комбинацией Ctrl-x Ctrl-s (C-x C-s) (рис. 4.4)

A screenshot of a terminal window. The prompt is 'U:---'. The current file is 'lab07.sh', and the editor is 'All L8 (Shell-script[sh])'. The message 'Wrote /home/sayprachanhlsnv/lab07.sh' is displayed on the line below the prompt.

```
U:--- lab07.sh All L8 (Shell-script[sh])
Wrote /home/sayprachanhlsnv/lab07.sh
```

Рис. 4.4: Сохранение файла

Вырезаю целую строку командой Ctrl-k (C-k) (рис. 4.5)

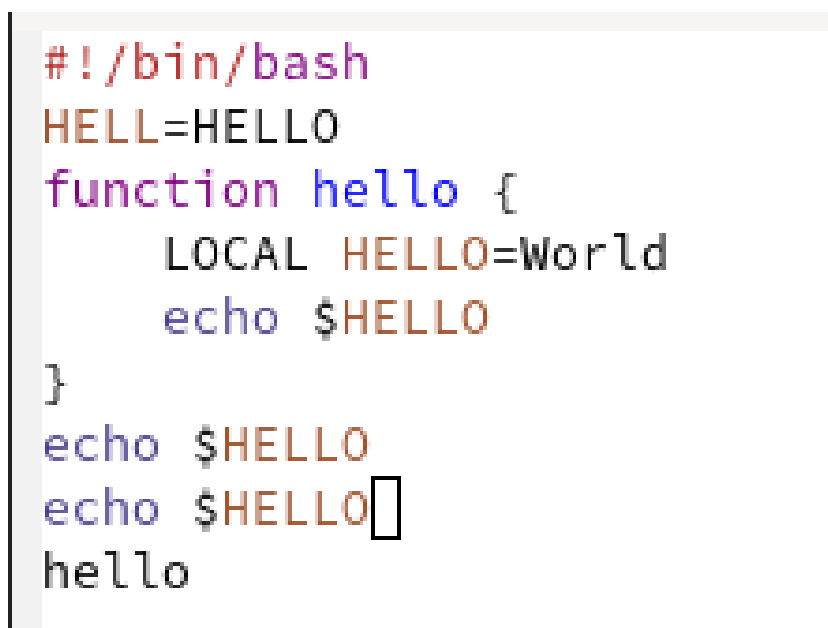


```
#!/bin/bash
HELL=HELLO
function hello {
    LOCAL HELLO=World
    echo $HELLO
}
echo $HELLO

```

Рис. 4.5: Вырезание целой линии

Пытаюсь вырезать другую строку, затем вставляю её в конец файла lab07.sh командой Ctrl-y (C-y) (рис. 4.6)



```
#!/bin/bash
HELL=HELLO
function hello {
    LOCAL HELLO=World
    echo $HELLO
}
echo $HELLO
echo $HELLO
hello

```

Рис. 4.6: Вставка строки

Для выделения области текста использую комбинацию Ctrl-space (C-space), затем копирую выделенный текст командой Alt-w или End-w (M-w) (рис. 4.7)

```
    echo $HELLO
}
echo $HELLO
echo $HELLO
hello
```

Рис. 4.7: Выделение текста и его копирование

Снова используя Ctrl-y (C-y), вставляю скопированный текст в конец файла (рис. 4.8)

```
#!/bin/bash
HELL=HELLO
function hello {
    LOCAL HELLO=World
    echo $HELLO
}
echo $HELLO
echo $HELLO
hello
echo $HELLO
```

Рис. 4.8: Вставка строки

Выделяю вставленный текст в конце файла и вырезаю его командой Ctrl-w (C-w) (рис. 4.9)

```

}
echo $HELLO
echo $HELLO
hello

```

Рис. 4.9: Вырезание целой линии

Отменяю предыдущее действие комбинацией Ctrl-/ (C-/), и мы видим, что текст возвращается к исходному состоянию до вырезания (рис. 4.10 и рис. 4.11)

```

U: *- lab07.sh All L10
Undo

```

Рис. 4.10: Отмена последнего действия

```

}
echo $HELLO
echo $HELLO
hello
echo $HELLO

```

Рис. 4.11: Текстовый код

Перемещаю курсор в начало строки командой Ctrl-a (C-a) (рис. 4.12)

```

echo $HELLO
hello
echo $HELLO

```

Рис. 4.12: Курсор в начале строки

Затем перемещаю курсор в конец строки комбинацией Ctrl-e (C-e) (рис. 4.13)

```
hello
echo $HELLO
```

Рис. 4.13: Курсор в конце строки

Перемещаю курсор в начало буфера (файла) с помощью Alt-< или Esc-< (M-<) (рис. 4.14)

```
#!/bin/bash
HELL=HELLO
function hello {
    LOCAL HELLO=World
```

Рис. 4.14: Курсор в начале файла

Далее, используя Alt-> или Esc-> (M->), перемещаю курсор в конец файла (рис. 4.15)

```
#!/bin/bash
HELL=HELLO
function hello {
    LOCAL HELLO=World
    echo $HELL
}
echo $HELL
echo $HELL
hello
echo $HELL
```

Рис. 4.15: Курсор в конце файла

4.2 Управление буферами

Отображаю список активных буферов на экране комбинацией Ctrl-x Ctrl-b (C-x C-b), затем переключаюсь между буфером lab07.sh и списком активных буферов командой Ctrl-x и буквой o (Ctrl-x o) (рис. 4.16)

CRM	Buffer	Size	Mode	File
□ *	lab07.sh	121	Shell-script[sh]	~/lab07.sh
%	*GNU Emacs*	718	Fundamental	
	scratch	145	Lisp Interaction	
%*	*Messages*	1590	Messages	
%*	*Async-native-compile-log*	165	Fundamental	

Рис. 4.16: Список активных буферов

Закрываю вкладку со списком буферов комбинацией Ctrl-x и цифрой 0 (Ctrl-x 0) (рис. 4.17)

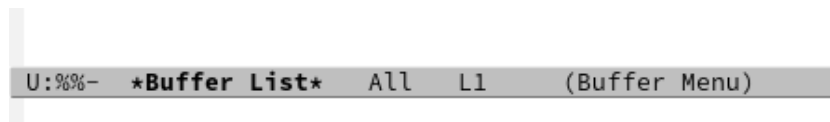


Рис. 4.17: Заккрытие буфера

Переключаюсь между буферами без отображения их списка на экране с помощью Ctrl-x и буквы b (C-x b), выбираю нужный буфер (Buffer List) (рис. 4.18 и рис. 4.19)

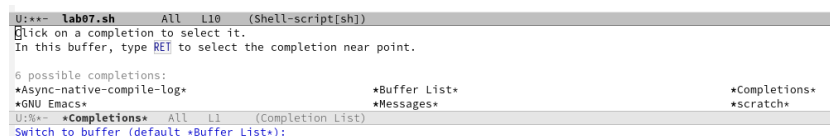


Рис. 4.18: Переключение на буфер

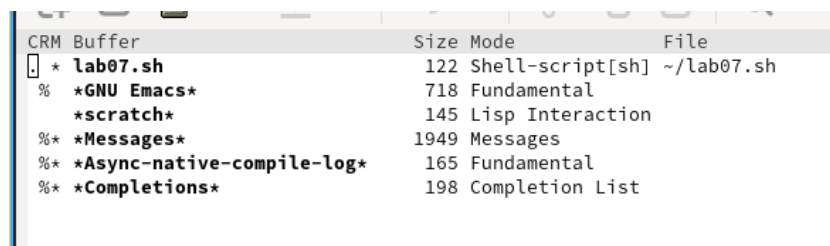


Рис. 4.19: Буфер Buffer List

4.3 Управление окнами

Возвращаюсь к буферу lab07.sh, затем разделяю фрейм на 4 части: комбинацией Ctrl-x с цифрой 3 (C-x 3) делю окно по вертикали, а с помощью Ctrl-x с цифрой 2 (C-x 2) — по горизонтали (рис. 4.20)



Рис. 4.20: Разделение буферного кадра

Открываю новый буфер в каждом из четырех созданных окон и ввожу в них текст (рис. 4.21)



Рис. 4.21: Создание новых буферов

4.4 Режим поиска

Переключаюсь в режим поиска командой Ctrl-s (C-s), нахожу несколько слов в тексте, также добавляю еще текст для дополнительного поиска (рис. 4.22 и рис. 4.23)



Рис. 4.22: Режим поиска

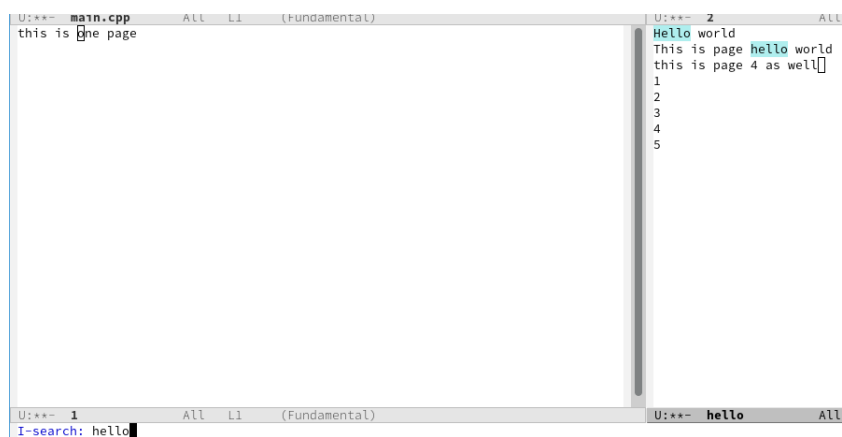


Рис. 4.23: Режим поиска

Переключаюсь между результатами поиска комбинацией Ctrl-S (C-S) (рис. 4.24)



Рис. 4.24: Переключение между результатами

Выхожу из режима поиска командой Ctrl-g (C-g) (рис. 4.25)



Рис. 4.25: Выход из режима поиска

Активирую режим поиска и замены комбинацией Alt-% или Esc-% (M-%), ввожу текст для поиска и замены, нажимаю Enter для применения изменений (рис. 4.26)

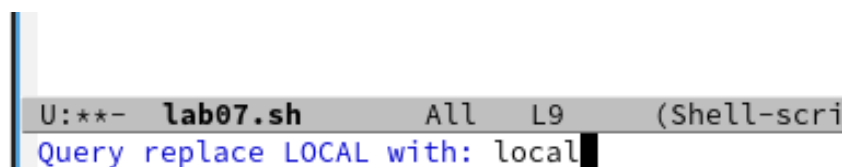


Рис. 4.26: Режим поиска и замены

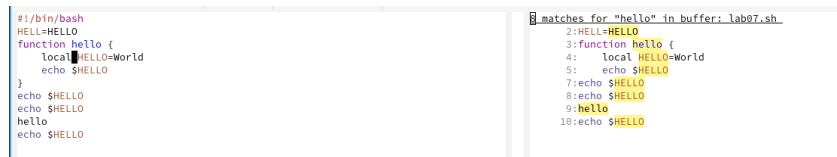
Проверяю результат — видно, что слово заменилось на новое (я заменил слово 'LOCAL' на 'local') (рис. 4.27)

```
#!/bin/bash
HELL=HELLO
function hello {
    local HELLO=World
    echo $HELLO
}
echo $HELLO
echo $HELLO
hello
echo $HELLO
```

U:***- lab07.sh All L4
Replaced 1 occurrence

Рис. 4.27: Замена слова

Пробую другой режим поиска: Alt-s или Esc-s с буквой o (M-s o), ввожу иско-
мый текст, и результаты отображаются в отдельном окне (в отличие от обычного
поиска) (рис. 4.28)



```
#!/bin/bash
HELL=HELLO
function hello {
    local HELLO=World
    echo $HELLO
}
echo $HELLO
echo $HELLO
hello
echo $HELLO
```

```
8 matches for "hello" in buffer: lab07.sh
2:HELL=HELLO
3:function hello {
4:    local HELLO=World
5:    echo $HELLO
7:echo $HELLO
8:echo $HELLO
9:hello
10:echo $HELLO
```

Рис. 4.28: Другой режим поиска

5 Выводы

Я познакомился с операционной системой Linux и получил практические навыки работы с редактором Emacs

6 Ответы на контрольные вопросы

1. Кратко охарактеризуйте редактор emacs.

Emacs — это мощный, расширяемый текстовый редактор с большим количеством функций, включая подсветку синтаксиса, работу с файлами, управление буферами, встроенный терминал, поддержку языков программирования и возможность кастомизации.

2. Какие особенности данного редактора могут сделать его сложным для освоения новичком?

Необычные сочетания клавиш (например, C-x C-s для сохранения).

Модальность (разные режимы ввода и команд).

Огромное количество функций, которые могут перегружать пользователя.

Настройка через Lisp (может быть сложной для непрограммистов).

3. Своими словами опишите, что такое буфер и окно в терминологии emacs'a.

Буфер — это объект Emacs, представляющий открытый файл, текстовые данные или процесс (например, терминал). Окно — это область на экране, в котором отображается буфер. В одном графическом окне Emacs может быть несколько внутренних окон (фреймов).

4. Можно ли открыть больше 10 буферов в одном окне?

Да, мы можем открыть столько буферов, сколько захотим, но только некоторые из них отображаются в одном окне за раз (в зависимости от разбивки на подокна).

5. Какие буферы создаются по умолчанию при запуске emacs?

При запуске создаются буферы:

scratch (временный буфер для тестирования кода)

Messages (лог системных сообщений)

GNU Emacs (справка или приветствие)

6. Какие клавиши вы нажмёте, чтобы ввести следующую комбинацию C-c | и C-c C-|?

Для ввода:

C-c | → Зажать Ctrl, нажать c, отпустить, затем нажать | (Shift +).

C-c C-| → Зажать Ctrl, нажать c, затем, не отпуская Ctrl, нажать |.

7. Как поделить текущее окно на две части?

Поделить окно:

Горизонтально: C-x 2 (Ctrl + x, затем 2).

Вертикально: C-x 3 (Ctrl + x, затем 3)

8. В каком файле хранятся настройки редактора emacs?

Настройки хранятся в файле ~/.emacs или ~/.emacs.d/init.el.

9. Какую функцию выполняет клавиша (<-) и можно ли её переназначить?

Клавиша <- (Backspace) обычно удаляет символ перед курсором. Её можно переназначить через настройки (например, в .emacs).

10. Какой редактор вам показался удобнее в работе vi или emacs? Поясните почему?

Emacs удобен для тех, кто любит кастомизацию, работу с разными режимами и Lisp.

Vi/Vim проще для быстрого редактирования благодаря модальности.

Если привыкнуть к сочетаниям клавиш, Emacs может быть удобнее из-за гибкости, но Vim быстрее для простого редактирования.

Список литературы

Лабораторная работа №11