

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ЯДЕРНЫЙ УНИВЕРСИТЕТ «МИФИ»

Лабораторный практикум

Тема: Основные способы работы с системой, основные команды.

Содержание

1	pwd	2
2	ls	2
3	mkdir	2
4	cd	2
5	Абсолютный путь и относительные пути	2
6	touch	4
7	echo	4
8	cat	4
9	cp	4
10	mv	5
11	file	5
12	whereis	5
13	Способы переместить файл f1.txt в другой каталог	5
14	rm	5
15	man	6
16	help	6
17	Стрелочки	6
18	Ctrl+r	6
19	history	6
20	less	6
21	Tab	6
22	nano/ micro/ (n)vim/ emacs	7
23	valgrind	7
24	tmux	8

1 pwd

print working directory - выводит полный путь от корневого каталога к текущему рабочему каталогу: в контексте которого будут исполняться вводимые команды. Позволяет узнать путь текущего рабочего каталога.

Знак «Тильда» - псевдоним домашнего каталога.

```
[test@unix:~]$ pwd
/home/test
```

2 ls

list sorted - используется для вывода содержимого каталогов и информации о файлах. В качестве аргументов ls принимает адреса каталогов или файлов. Если их не передавать, то будет выведено содержимое текущего каталога.

```
[test@unix:~]$ ls
```

3 mkdir

make directory - команда для создания новых каталогов.

```
[test@unix:~]$ mkdir d1
```

После создания каталога d1, можно увидеть ее наличие с помощью команды ls:

```
[test@unix:~]$ ls
d1
```

4 cd

change directory - используется для изменения текущего рабочего каталога.

```
[test@unix:~]$ cd d1
[test@unix:~/d1]$
```

После выполнения команды можно заметить, что после знака тильды появилось «d1». (« » - то же самое, что и /home/test). Таким образом, мы погрузились вниз на один каталог.

Если еще раз применим pwd, можно увидеть каким образом устроен абсолютный путь текущего каталога:

```
[test@unix:~/d1]$ pwd
/home/test/d1
[test@unix:~/d1]$
```

5 Абсолютный путь и относительные пути

У любого файла (каталог – это частный случай файла) есть один абсолютный путь и множество относительных путей. Абсолютный путь показывает точное местонахождение файла, а относительный показывает путь к файлу относительно какой-либо "отправной точки" (файл, программа и т. д.). Абсолютный путь начинается с корневого каталога.

```
[test@unix:~]$ pwd  
/home/test
```

/home/test

- каталогом верхнего уровня является корень, который мы обозначаем знаком «/» в самом начале, внутри него находится каталог «home», а внутри него – каталог «test».

/home/test

- следующий «/» - разделитель, разделяющий имена каталогов.

Существует некий стандарт, который оговаривается стандартом POSIX, о том, как должны выглядеть каталоги в POSIX совместимых операционных системах. То, что мы видим – некое приближение к ним:

```
[test@unix:/]$ ls  
bin    dev    home    nix    proc    run    sys    usr  
boot   etc    lost+found  opt    root    srv    tmp    var
```

Эта система каталогов она более-менее стандарта. Вся система каталогов представляет из себя единую древовидную систему. Вершиной является корневой каталог.

```
[test@unix:/]$ cd /home/  
[test@unix:/home]$ ls  
test  
[test@unix:/home]$ tree  
.  
└─ test  
    └─ d1  
2 directories, 0 files
```

знак "Тильда псевдоним домашнего каталога

. – текущий каталог

.. – каталог на уровень выше

*подробнее можно изучить в книге «Операционная система UNIX» Андрей Робачевский.

И если мы перейдем в корневой каталог и попробуем перейти выше – останемся в корневом. К примеру, перейдем в d1 и рассмотрим способы вернуться в домашний каталог.

```
[test@unix:~]$ cd d1/  
[test@unix:~/d1]$ pwd  
/home/test/d1
```

Способы вернуться в домашний каталог:

a. [test@unix:~/d1]\$ cd ..

b. [test@unix:~/d1]\$ cd ~

c. [test@unix:~/d1]\$ cd /home/test/

d. [test@unix:~/d1]\$ cd ../../d1/../../test/

- т.о., у нас бесконечное количество относительных путей (любой путь, начинающийся не с вершины).

6 touch

Создает файл, обновляет временную метку.

К примеру, создадим файл f1: `[test@unix:~/d1]$ touch f1`

И так как ранее такого файла не было, то такой файл будет создан и его временная метка будет равна текущему времени:

```
[test@unix:~/d1]$ ls
f1
[test@unix:~/d1]$ stat f1
  File: f1
  Size: 0          Blocks: 0          IO Block: 4096
regular empty file
Device: 253,2    Inode: 1179653    Links: 1
Access: (0644/-rw-r--r--)  Uid: ( 1001/   test)   Gid: (
100/   users)
Access: 2022-09-14 20:34:31.314922969 +0300
Modify: 2022-09-14 20:34:31.314922969 +0300
Change: 2022-09-14 20:34:31.314922969 +0300
 Birth: 2022-09-14 20:34:31.314922969 +0300
```

Попробуем изучить еще несколько команд для создания файлов:

7 echo

Существует команда echo для вывода текста в стандартном потоке вывода. К примеру, создадим файл f1:

```
[test@unix:~/d1]$ echo Hello
Hello
```

Можем перенаправить стандартный поток выхода на ввод другой команды с помощью конвейера или в другой файл с помощью значка «больше (>)»:

```
[test@unix:~/d1]$ echo Hello > f1.txt
[test@unix:~/d1]$ ls
f1  f1.txt
```

Теперь у нас есть 2 файла: f1 и f1.txt.

8 cat

Посмотрим содержимое файлов:

```
[test@unix:~/d1]$ cat f1.txt
Hello
[test@unix:~/d1]$ cat f1
[test@unix:~/d1]$
```

f1 пуст и f1.txt содержит текст.

9 cp

copy — копирование

```
[test@unix:~/d1]$ cp f1.txt f2.txt
```

10 mv

move – перемещение

```
[test@unix:~/d1]$ mv f2.txt f3
```

Скопировали сначала f1.txt в f2.txt и потом переименовали f2.txt в f3.txt. Теперь можно видеть следующее:

```
[test@unix:~/d1]$ cat f1.txt
Hello
```

```
[test@unix:~/d1]$ cat f3
Hello
```

Эти файлы отличаются теперь только именем.

11 file

Утилита file позволяет получать информацию о файле.

```
[test@unix:~/d1]$ file f1.txt
f1.txt: ASCII text
```

12 whereis

Позволяет найти расположение двоичных файлов, файлов исходного кода и файлов справочной страницы для данной команды. Т.о., можем увидеть, где находится конкретный исполняемый файл.

```
[test@unix:~/d1]$ whereis mount
mount: /run/wrappers/wrappers.30WwXQZFz/mount /nix/store/
3y8slhszlnavqkscy2kcz5dis3pyvm-system-path/bin/mount
```

13 Способы переместить файл f1.txt в другой каталог

Мы можем менять расположение файлов из любого другого каталога, без необходимости менять текущий каталог.

a.

```
[test@unix:~/d1]$ mv f1.txt /tmp
```

b.

```
[test@unix:~/d1]$ mv /tmp/f1.txt .
```

14 rm

rm - remove – удаление из файловой системы. rmdir – remove directory – удаление пустого каталога из файловой системы.

(ключи -r -f (можно записать как в примере выше « -rf ») : r – рекурсивное удаление, f – без подтверждения).

```
[test@unix:~/d1]$ ls
f1 f1.txt f3

[test@unix:~/d1]$ rm f1

[test@unix:~/d1]$ cd ..

[test@unix:~]$ rm -rf d1/
```

15 man

manual- справки по внешним командам. rmdir – remove directory – удаление пустого каталога из файловой системы.

```
[test@unix:~]$ man rmdir
```

16 help

Справки по внутренним командам (встроенным в интерпретатор).

```
[test@unix:~]$ rmdir --help
```

17 Стрелочки

Стрелочки вверх – вниз позволяют переключаться между командами, которые были введены.

18 Ctrl+r

Реверсивный поиск позволяет значительно ускорить работу. С помощью клавиш Ctrl+r можно начать вбивать любую команду, которой уже пользовались (переключаемся по истории команд каждым нажатием Ctrl+r).

19 history

history - дает возможность смотреть историю всех выполненных команд. history -c удаляет всю историю.

20 less

less используется для просмотра (но не изменения) содержимого текстовых файлов на экране.

21 Tab

Кнопка «Tab» – автодополнение - по нажатию клавиши Tab дописывается название команды (на столько, на сколько возможно). А по второму нажатию Tab отображается список всех доступных вариантов. К примеру, на «log»:

```
[test@unix:~]$ log
logger  loginctl  logout  logsave
login   logname   logoutd
```

22 nano/ micro/ (n)vim/ emacs

nano/ micro/ (n)vim/ emacs – текстовые редакторы для использования на сервере.

a. Редактор nano: `[test@unix:~/d1]$ nano f1`

Сохранение и выход - Ctrl+O, Ctrl+x .

b. Редактор micro: `[test@unix:~/d1]$ micro f1`

Позволяет использовать все те сочетания клавиш, что используются в редакторах Windows. После ввода текста нажимаем Ctrl+s. Для выхода Ctrl+q.

c. Редактор (n)vim: `[test@unix:~/d1]$ nvim hello.c`

Напишем простой «Hello»



:wq и Enter для сохранения и выхода.

Скомпилируем код с помощью компилятора gcc, который можем вызывать из команды

```
[test@unix:~/d1]$ cc -o abc hello.c
```

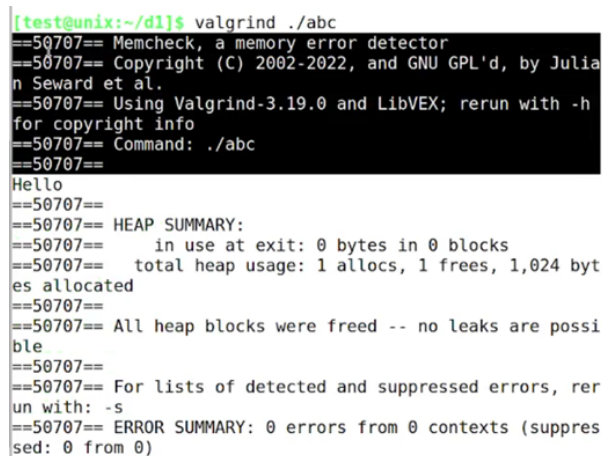
-o – модификатор, с которого можно задать имя выходного файла;

На выходе получаем исполняемый файл hello:

```
[test@unix:~/d1]$ ./abc
Hello
```

23 valgrind

valgrind известен как средство поиска ошибок работы с памятью. Можно проанализировать выделение и освобождение памяти.

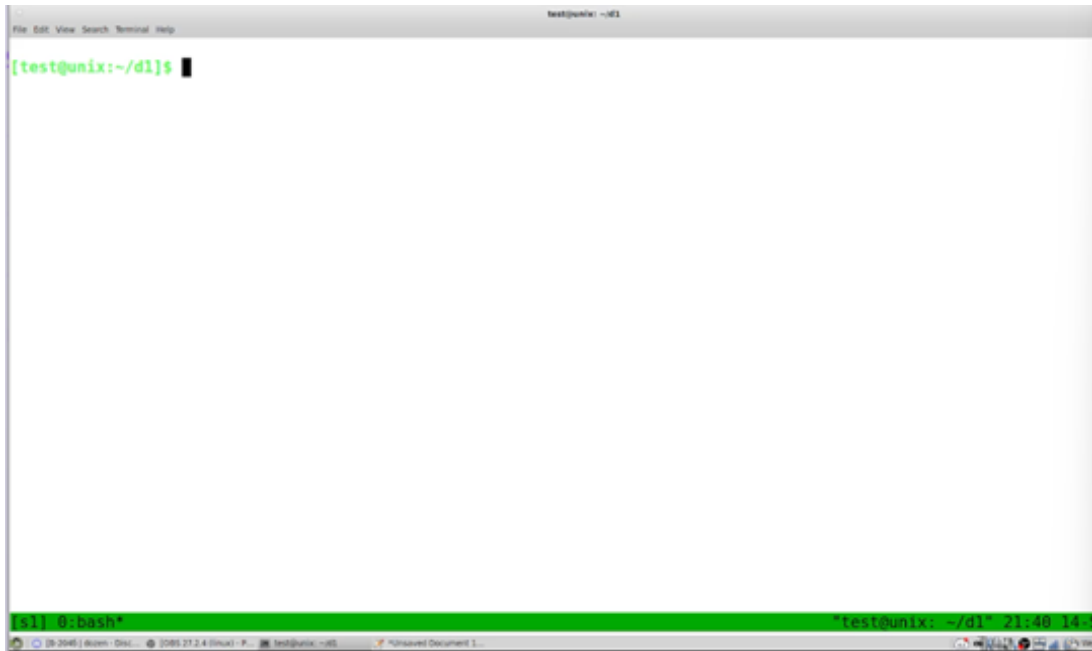


24 tmux

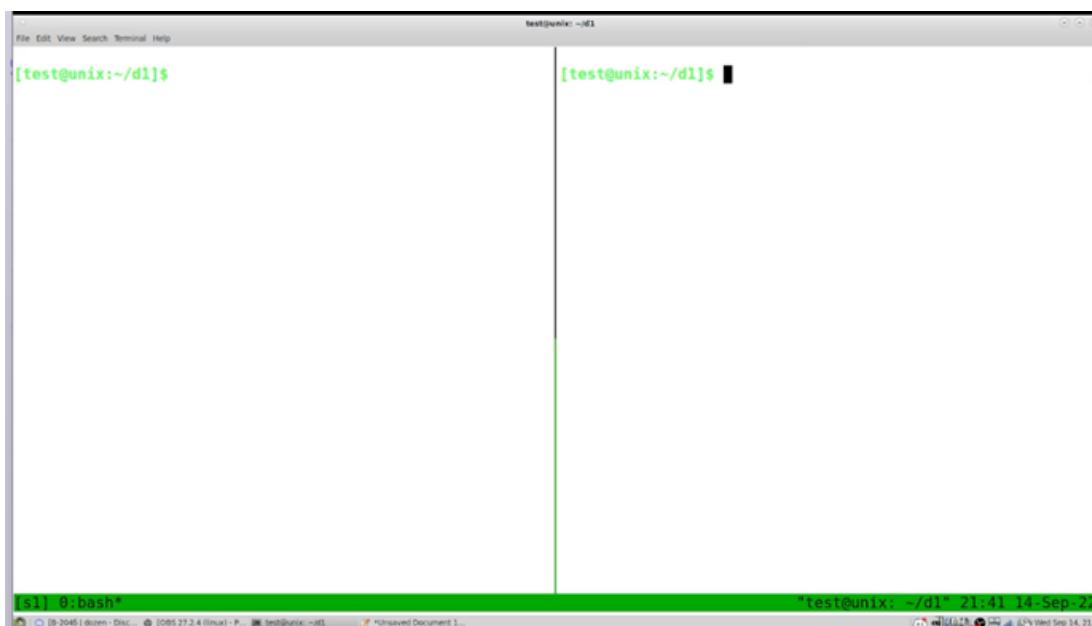
tmux (терминальный мультиплексор) позволяет работать с несколькими сессиями в 1 окне; утилита-мультиплексор, предоставляющая пользователю доступ к нескольким терминалам в рамках одного экрана.

```
[test@unix:~/d1]$ tmux new -s s1
```

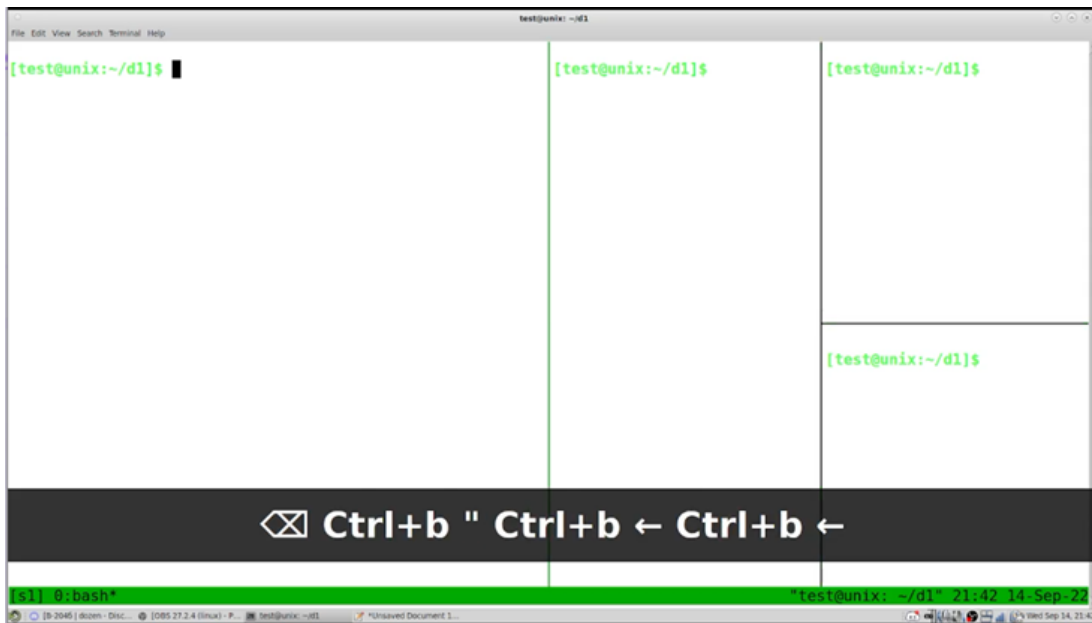
После создания новой сессии внизу появилась зеленая строка.



Для горизонтального разделения окна используется сочетание клавиш Ctrl + b, “ (Shift + ‘(э)). Чтобы разделить окно вертикально на две равные панели — воспользуйтесь сочетанием клавиш Ctrl + b, % (Shift + 5 (знак процента)). Перемещаться между панелями можно с помощью сочетаний клавиш Ctrl + b и стрелок.



Ctrl+b x — удаляем панель. Ctrl+b (Shift + 7 (Амперсанд)) закрывает окно. После закрытия всех окон происходит выход из сессии.



Переключиться между окнами можно с помощью следующих сочетаний клавиш:

- **Ctrl + b, n** — следующее окно
- **Ctrl + b, p** — предыдущее окно
- **Ctrl + b, w** — следующее окно
- **Ctrl + b, номер окна (цифрой)** — переключиться на нужное окно

Список часто используемых команд и хоткейсов Tmux

Команды для управления сессиями:

- **tmux new [имя_сеанса]** — начать новый сеанс. Имя_сеанса опционально;
- **tmux attach -t [имя_сеанса]** - подключиться к уже существующей сессии. Если имя заранее не было задано, тогда команда будет выглядеть так: **tmux attach -t 0**;
- **tmux ls** — список открытых сессий Tmux;
- **kill-server** — остановить все запущенные сессии;
- **kill-session -t [имя_сеанса]** — завершить сессию;
- **list-clients -t [имя_сеанса]** — посмотреть клиентов, подключенных к сессии;
- **list-sessions** — вывести список всех запущенных сессий.

Хоткейсы для управления окнами:

- **Ctrl + b, c** — создать новое окно;
- **Ctrl + b, w** — просмотреть список окон;
- **Ctrl + b, n** — следующее окно;
- **Ctrl + b, p** — предыдущее окно;
- **Ctrl + b, w** — следующее окно;
- **Ctrl + b, номер окна (цифрой)** — переключиться на нужное окно;
- **Ctrl + b, "** — горизонтальное разделение окна;
- **Ctrl + b, %** — вертикальное разделение окна.