

Рафи Кази ар

НКАбд-03-24

1032238132@pfur.ru

Кулябов Дмитрий Сергеевич

Moscow

6.3. Последовательность выполнения работы.

Решение заданий лабораторной работы

Примечание: Все команды предполагается выполнять в терминале Linux. Замените username на ваше реальное имя пользователя.

1. Осуществите вход в систему.

- Это действие выполняется на этапе загрузки ОС или при подключении к серверу по SSH.

2. Запись названий файлов в file.txt

Записываем (перезаписываем) список файлов из /etc в file.txt

```
ls /etc > file.txt
```

Дописываем (append) список файлов из домашнего каталога в тот же file.txt

```
ls ~ » file.txt
```

3. Поиск .conf файлов и запись в conf.txt

Ищем в file.txt строки, содержащие ".conf", и записываем результат в conf.txt

```
grep "\.conf$" file.txt > conf.txt
```

- \. - экранирование точки (чтобы она означала именно точку, а не любой символ).
- \$ - означает "конец строки", так мы ищем именно расширение.

4. Поиск файлов, начинающихся на 'с' в домашнем каталоге Вариант 1 (с помощью ls):

```
ls ~/с*
```

Вариант 2 (с помощью find - более надежный):

```
find ~ -maxdepth 1 -name "с*" -type f
```

- -maxdepth 1 - искать только в домашнем каталоге, не углубляясь в подкаталоги.

Вариант 3 (с помощью конвейера и grep):

```

[❏ krafi@pop-os ] [❏ 09:19 PM] [❏ www.krafi.
└─❏ /tmp
➔ ls /etc > file.txt

[❏ krafi@pop-os ] [❏ 09:19 PM] [❏ www.krafi.
└─❏ /tmp
➔ ls ~ >> file.txt

[❏ krafi@pop-os ] [❏ 09:19 PM] [❏ www.krafi.
└─❏ /tmp
➔ grep "\.conf$" file.txt > conf.txt

[❏ krafi@pop-os ] [❏ 09:19 PM] [❏ www.krafi.
└─❏ /tmp
➔ ls ~/c*
zsh: no matches found: /home/krafi/c*

[❏ krafi@pop-os ] [❏ 09:20 PM] [❏ www.krafi.
└─❏ /tmp
➔ find ~ -maxdepth 1 -name "c*" -type f

[❏ krafi@pop-os ] [❏ 09:20 PM] [❏ www.krafi.
└─❏ /tmp
➔ ls ~ | grep "^c"

[❏ krafi@pop-os ] [❏ 09:20 PM] [❏ www.krafi.
└─❏ /tmp
➔ ls /etc/h* | less
ls ~ | grep "^c"

```

5. Постраничный вывод файлов из /etc, начинающихся на 'h'

ls /etc/h* | less

```
/etc/host.conf
/etc/hostname
/etc/hosts
/etc/hosts.allow
/etc/hosts.deny
(END)
```

или

```
find /etc -maxdepth 1 -name "h*" -print | less
```

**6. Запуск фонового процесса поиска файлов log*

```
find / -name "log*" > ~/logfile 2> /dev/null &
```

- 2> /dev/null - перенаправляет ошибки (например, "Permission denied") в "никуда", чтобы они не засоряли вывод.
- & - запуск команды в фоновом режиме.

7. Удаление файла ~/logfile

```
rm ~/logfile
```

8. Запуск gedit в фоновом режиме

```
gedit &
```

- После запуска вы увидите примерно: [1] 12345, где 1 - номер задачи (job), 12345 - PID процесса.

9. Определение PID процесса gedit Способ 1 (через ps и grep):

```
[krafi@pop-os ] 09:21 PM > www.kra
└─ /tmp
> find / -name "log*" > ~/logfile 2> /dev/null &
[1] 129716

[krafi@pop-os ] 09:21 PM > www.kra
└─ /tmp
> rm ~/logfile

[krafi@pop-os ] 09:21 PM > www.kra
└─ /tmp
> gedit &
[2] 129760

[krafi@pop-os ] 09:21 PM > www.kra
└─ /tmp
[2] + 129760 done gedit
```

ps aux | grep gedit

- В выводе вы увидите строку с процессом gedit и его PID (второй столбец).

Способ 2 (через pgrep):

pgrep gedit

- Эта команда выведет только PID процесса.

Способ 3 (через jobs -l):

jobs -l

- Команда покажет все задачи, запущенные из текущей оболочки, с их номерами и PID.

10. Завершение процесса gedit с помощью kill

- Сначала смотрим справку: man kill
- Основной синтаксис: kill [опции] <PID>
- Завершаем процесс (подставив реальный PID, например, 12345):

kill 12345

- Если процесс не реагирует, можно использовать `kill -9 12345` (более жесткое, безусловное завершение).

11. Выполнение команд `df` и `du`

- Читаем справку: `man df`, `man du`
- Выполняем команды с полезными опциями:

`df -h` # Размеры разделов в удобном формате (G, M)

```
[krafia@pop-os ~]$ pgrep gedit
12345

[krafia@pop-os ~]$ jobs -l
^C

[krafia@pop-os ~]$ kill 12345
kill: kill 12345 failed: no such process

[krafia@pop-os ~]$ df -h
Filesystem                Size      Used Avail Use% Mounted on
tmpfs                      1.4G    2.4M    1.4G   1% /
efivarfs                   128K     53K     71K  43% /
/dev/mapper/my--vg-root    200G     46G    154G  23% /
tmpfs                      6.8G    182M    6.6G   3% /
tmpfs                      5.0M      0    5.0M   0% /
/dev/mapper/my--vg-home    281G    192G     86G  70% /
/dev/nvme0n1p1             1020M    210M    811M  21% /
/dev/nvme0n1p2             4.0G     3.0G    1.1G  75% /
tmpfs                      1.4G     1.8M    1.4G   1% /

[krafia@pop-os ~]$ du -sh
du: cannot read directory './snap-private-tmp':
du: cannot read directory './systemd-private-959
du: cannot read directory './systemd-private-959
du: cannot read directory './systemd-private-959
du: cannot read directory './systemd-private-959
du: cannot read directory './systemd-private-959
du: cannot read directory './systemd-private-959
du: cannot read directory './systemd-private-959
du: cannot read directory './systemd-private-959
du: cannot read directory './systemd-private-959
du: cannot read directory './systemd-private-959
100K
```

`du -sh ~/` # Суммарный размер домашнего каталога

12. Поиск всех директорий в домашнем каталоге

```
man find
FIND(1)                                General Commands Manual

NAME
    find - search for files in a directory hierarchy

SYNOPSIS
    find [-H] [-L] [-P] [-D debugopts] [-Olevel] [starting-point...] [expression]

DESCRIPTION
    This manual page documents the GNU version of find. GNU find searches for files by evaluating the given expression from left to right, according to the rules of Boolean algebra. The outcome is known (the left hand side is false for and operations, true for or operations). If no starting-point is specified, . is assumed.

    If you are using find in an environment where security is important (for example, that are writable by other users), you should read the 'Security Considerations' section which is called Finding Files and comes with findutils. That document also contains this manual page, so you may find it a more useful source of information.

OPTIONS
    The -H, -L and -P options control the treatment of symbolic links. -H follows names of files or directories to be examined, up to the first argument that is a symbolic link. That argument and any following arguments are taken to be the expression to be evaluated. If no expression is given, the current directory is used. If no expression is given, then consider using -print0 instead, anyway).

    This manual page talks about 'options' within the expression list. These options are specified immediately after the last path name. The five 'real' options -H, -L, -P, -type and -name, if at all. A double dash -- could theoretically be used to signify the end of the options, but this does not really work due to the way find determines the end of the expression list. Until an expression argument comes (which also starts with a '). Now, find would treat it as expression argument instead. Thus, to ensure that all options are recognized, it is safer to prefix wildcards or dubious path names with either ./ or ../.

    -P Never follow symbolic links. This is the default behaviour. When find examines or prints information about a file and the file is a symbolic link, the information used shall be taken from the file itself, not from the file to which the link points.

    -L Follow symbolic links. When find examines or prints information about a file, the properties of the file to which the link points, not from the file itself, are used. find is unable to examine the file to which the link points. If the -P option, -noleaf will still be in effect. If -L is in effect, the subdirectory pointed to by the symbolic link is examined.

    When the -L option is in effect, the -type predicate will always return d if the link points to rather than the link itself (unless the symbolic link is broken). If the link becomes broken while find is executing (for example -delete causes the -lname and -ilname predicates always to return false.

    -H Do not follow symbolic links, except while processing the command line. When find examines or prints information about files, the information used shall be taken from the file itself, not from the file to which the link points. The exception to this behaviour is when a file specified on the command line is a symbolic link. For that situation, the information used is taken from the file to which the link points (unless the symbolic link is broken). The information about the link itself is used as a fallback. If the file is not examined. If -H is in effect and one of the paths specified on the command line is a symbolic link, the contents of that directory will be examined (though of course, the file to which the link points will not be examined).

    If more than one of -H, -L and -P is specified, each overrides the other. Since it is the default, the -P option should be considered to be in effect unless overridden.

    GNU find frequently stats files during the processing of the command line.

Manual page find(1) line 1 (press h for help or q to quit)
```

- Читаем справку: `man find`
- Выполняем поиск:
`find ~ -type d`
- `-type d` - критерий поиска: найти только директории (d).

Ответы на контрольные вопросы (6.5)

1. **Какие потоки ввода вывода вы знаете?**
 - `stdin` (0) - стандартный поток ввода (по умолчанию: клавиатура).
 - `stdout` (1) - стандартный поток вывода (по умолчанию: экран).
 - `stderr` (2) - стандартный поток ошибок (по умолчанию: экран).
2. **Объясните разницу между операцией `>` и `»`.**
 - `>` - перенаправление вывода с **перезаписью** файла. Если файл существует, он будет стерт и создан заново.
 - `»` - перенаправление вывода с **добавлением** в конец файла. Если файл существует, данные будут дописаны после существующих.
3. **Что такое конвейер?**
 - Конвейер (`|`) - это механизм, который передает стандартный вывод (`stdout`) одной команды на стандартный ввод (`stdin`) другой команды. Позволяет объединять простые команды в мощные цепочки.
4. **Что такое процесс? Чем это понятие отличается от программы?**
 - **Программа** - это набор инструкций и данных, хранящийся на диске (исполняемый файл).
 - **Процесс** - это экземпляр программы, **выполняющийся в памяти**. Он имеет свой собственный адресное пространство, ресурсы системы и идентификатор (PID). Одна программа может быть запущена в виде нескольких процессов.
5. **Что такое PID и GID?**

- **PID (Process ID)** - уникальный числовой идентификатор процесса в системе.
- **GID (Group ID)** - идентификатор группы. Каждый пользователь и процесс принадлежит к одной или нескольким группам. **PPID (Parent Process ID)** - идентификатор процесса-родителя, который создал данный процесс.

6. **Что такое задачи и какая команда позволяет ими управлять?**

- **Задачи (jobs)** - это процессы, запущенные в **фоновом режиме** из текущей оболочки (сессии терминала).
- Управлять ими позволяет команда `jobs` (показать список), а также:
 - `fg %N` - вернуть задачу на передний план.
 - `bg %N` - запустить остановленную задачу в фоне.
 - `kill %N` - завершить задачу.

7. **Найдите информацию об утилитах `top` и `htop`.**

- **top** - интерактивная консольная команда для мониторинга системы в реальном времени. Показывает список процессов, использование CPU, памяти, время работы и другую информацию.
- **htop** - улучшенная версия `top`. Имеет более удобный интерфейс с цветовым выделением, позволяет управлять процессами (завершать, менять приоритет) с помощью клавиш, имеет горизонтальную и вертикальную прокрутку.

8. **Назовите и дайте характеристику команде поиска файлов.**

- **find** - мощнейшая команда для поиска файлов и каталогов по различным критериям: имя, размер, время изменения, тип файла,

владелец и т.д. Может выполнять действия над найденными файлами (удалить, показать и др.).

- **Примеры:**

- `find /home -name "*.txt"` - найти все .txt файлы в /home.
- `find . -size +10M` - найти файлы больше 10 МБ в текущей директории.
- `find /var/log -mtime -7` - найти файлы в /var/log, измененные за последние 7 дней.

9. **Можно ли по контексту (содержанию) найти файл? Если да, то как?**

- **Да, можно.** Для этого используется команда `grep` в сочетании с рекурсивным обходом.
- **Команда:** `grep -r "искомый_текст" /путь/для/поиска`
- **Пример:** `grep -r "Hello World" ~/projects/`
- найдет все файлы в каталоге `~/projects/`, внутри которых есть строка "Hello World".

10. **Как определить объем свободной памяти на жёстком диске?**

- С помощью команды `df (disk free)`.
- **Лучший вариант:** `df -h` - покажет размеры всех смонтированных разделов, занятое и свободное место в удобном для человека формате (Гиб, Миб).

11. **Как определить объем вашего домашнего каталога?**

- С помощью команды `du (disk usage)`.
- **Команда:** `du -sh ~/`
- `-s` - показать только итоговую сумму.
- `-h` - вывести размер в человеко-читаемом формате.

12. **Как удалить зависший процесс?**

1. Найти PID зависшего процесса (например, firefox): `pgrep firefox` или `ps aux | grep firefox`.
2. Попыаться завершить его корректно: `kill <PID>`.
3. Если не помогает, принудительно завершить: `kill -9 <PID>` (сигнал SIGKILL, который нельзя перехватить или проигнорировать).