ALTADM1.

Администрирование ОС Альт.

Часть 1

Практикум

МОСКВА 2023

Оглавление

Лабораторная работа 1

Задание 1.1

1. Результат выполнения задания — это **три** сценария на языке командного интерпретатора.:
   первый (основной) выполняет задание;
   второй (тестовый) проверяет правильность работы основного;
   третий (обнуляющий) возвращает систему в исходное состояние так, чтобы можно было запустить процесс выполнения сценариев сначала.

2. Задача **основного** сценария: создать файл **my_file**, записать в него с помощью соответствующих команд:
   системную дату,
   текущий рабочий каталог,
   имя пользователя,
   название и версию ядра операционной системы.

3. **Тестовый** сценарий должен проверить наличие файла **my_file** и вывести его содержимое.

4. **Третий** сценарий удаляет файл **my_file** или сообщает об его отсутствии.

5. Для начала работы запустите текстовый редактор **nano**. Наберите шебанг и первые комментарии с описанием сценария:
   ```
   #!/bin/sh
   # сценарий записи системной информации в файл
   # лабораторная работа № 1
   ```

   Дальше следуют команды создания файла **my_file** и записи данных в него:

   ```
   touch my_file

   date > my_file
   pwd >> my_file
   who >> my_file
   uname -sr >> my_file
   ```

6. Сохраните файл, нажав **^+X** (Ctrl+X).
   Имя файла первого скрипта первой работы должно быть **base1.sh**

7. Для того, чтобы можно было запустить скрипт, необходимо сделать файл исполняемым (наберите в командной строке, после выхода из редактора):
   ```
   chmod +x ./base1.sh
   ```

8. Попытайтесь выполнить сценарий:
   `./base1.sh`

9. Отобразите на экране содержимое файла **my_file**, выполнив:
   `cat my_file`

10. Вернитесь к редактированию файла **base1.sh**, набрав:
    `nano base1.sh`

11. Обратите внимание на цветное оформление сценария.
    Добавьте комментарии в строках создания и записи файла:

```
touch my_file  # создание файла

date > my_file    # запись даты и времени
pwd >> my_file    # запись текущего каталога
who >> my_file    # запись имени пользователя
uname >> my_file  # запись названия ОС
```

12. Сохраните изменения и закройте редактор.

13. Теперь создайте **тестовый** скрипт командой:
    `nano test1.sh`

    Так же начните с шебанга и описания скрипта:

```
#!/bin/sh
# сценарий проверки правильности выполнения
# Лабораторной работы № 1

cat my_file # вывод на  экран содержимого файла my_file
echo $?      # вывод кода завершения последней операции
```

14. Сохраните файл и закройте редактор.

15. Настройте права для выполнения и этого сценария:
    `chmod +x ./test1.sh`

16. Выполните сценарий:
    `./test1.sh`

17. Убедитесь, что код завершения (последнее число) равен **0**.
    Это говорит об **отсутствии ошибок** при выполнении команды.

18. Скопируйте файл **`test1.sh`** в новый файл **`reset1.sh`**. Затем вызовите его на редактирование:

```
cp test1.sh  reset1.sh

nano reset1.sh
```

19. Измените комментарии и содержание редактируемого файла:

```
#!/bin/sh
# сценарий очистки результатов выполнения задания

# удаление файла my_file
rm my_file && echo 'рабочий файл удалён'
echo $?         # вывод кода завершения последней операции
```

20. Сохраните файл и завершите редактирование. Как в предыдущих случаях сделайте файл исполняемым и запустите его:

```
chmod +x ./reset1.sh
./reset1.sh
```

21. Можно проверить все три файла ещё одним последовательным запуском, а затем сверить с образцом содержание самих скриптов:

base1.sh	#!/bin/sh # сценарий записи системной информации в файл # Задание 1.1 **touch my_file** # создание файла **date > my_file** # запись даты и времени **pwd >> my_file** # запись текущего каталога в файл **who >> my_file** # запись имени в файл **uname >> my_file** # запись названия ОС в файл
test1.sh	#!/bin/sh # сценарий проверки правильности выполнения # Задание 1.1 **cat my_file** # вывод на экран содержимого файла my_file **echo $?** # вывод кода завершения последней операции
reset1.sh	#!/bin/sh # сценарий очистки результатов выполнения # Задание 1.1 **rm my_file && echo 'рабочий файл удалён'** # удаление файла my_file **echo $?** # вывод на экран кода завершения последней операции

Задание 1.2

1. По аналогии с Заданием 1.1 сформируйте сценарий **base1.2.sh**. Сценарий должен выполнять следующие действия:

- с помощью команды **echo** и конструкции перенаправления вывода потока записать в файл **my_file** первую **строку, содержащую фамилию и имя слушателя** (по-английски в нижнем регистре).
- второй строкой добавить в файл **текущую дату и время**, используя перенаправление вывода команды **date**.
- третьей строкой добавить в файл **слово “date”**, используя команду **echo** и перенаправление текстового вывода.

2. Создайте сценарий **test1.2.sh** в котором с помощью утилиты grep отфильтруйте содержимое файла “my_file” и отобразите **строку, содержащую слово “date”**.
3. Подготовьте сценарий **reset1.2.sh**, выполняющий удаление из системы файла **my_file**

Задание 1.3 (Дополнительно)

В качестве дополнительного самостоятельного задания разработайте три аналогичных сценария, при этом в файл **my_hist_file** должно быть записано 10 последних команд текущего сеанса работы.

Лабораторная работа 2

Задание 2.1

1. Результат выполнения задания — это **три** сценария на языке командного интерпретатора.:
   первый (основной) выполняет задание;
   второй (тестовый) проверяет правильность работы основного;
   третий (обнуляющий) возвращает систему в исходное состояние так, чтобы можно было запустить процесс выполнения сценариев сначала.

2. Задача **основного** сценария задания 2.1:
   создать в **/tmp** в новом файле **my_sys_2M.img** размером **2M** файловую систему **ext2**;
   провести монтирование новой ФС в **/mnt/new_disk**;
   исследовать возможность заполнения ФС в двух вариантах:
   по **количеству** файлов;
   по **объёму** ФС (Задание 2.2);

3. Тестовый сценарий должен проверить наличие ФС, показать её параметры, количество созданных файлов.

4. Для возврата к начальным условиям необходимо очистить и отмонтировать файловую систему, а затем удалить файл **my_sys_2M.img** .

5. Для начала работы запустите текстовый редактор **nano**:
   nano base2.sh

6. Наберите шебанг и первые комментарии с описанием сценария:

```
#!/bin/sh
# сценарий создания и монтирования ФС
# Задание 2.1
```

Дальше следуют команды подготовки файла **my_sys_2M.img** и создания на нем ФС:

```
truncate -s 2M /tmp/my_sys_2M.img    # файл заданного размера

mkfs.ext2 /tmp/my_sys_2M.img                  # созд. ФС ext2
mkdir /mnt/new_disk                 # созд.точку монтирования
mount /tmp/my_sys_2M.img /mnt/new_disk         # монтируем ФС
df -H /mnt/new_disk                   # выводим сведения о ФС
```

Теперь приступаем к заполнению ФС:

```
mkdir /mnt/new_disk/new_dir      # создаем каталог в новой ФС
```

```
# в новом каталоге: сколько влезет файлов,прячем сообщения об ошибках
cd /mnt/new_disk/new_dir
i=1; while touch file_$i; do echo file_$i; i=$[i+1];done
```

(Предложите альтернативный вариант цикла, например с использованием **for**.)

7. Сохраните файл и закройте редактор. Настройте права на исполнение и запустите сценарий с правами суперпользователя. После выполнения не забудьте завершить сеанс работы суперпользователя.

```
chmod +x ./base2.sh
su -
cd <dir>*
./base2.sh
```

```
* - вернуться в каталог где создавали скрипт
```

8. Начните работу с тестовым сценарием:

```
nano test2.sh
```

9. Наберите шебанг и первые комментарии с описанием сценария:

```
#!/bin/sh
# сценарий исследования ФС
# Задание 2.1
```

10. Тестовый сценарий должен вывести информацию о файловой системе и посчитать количество файлов в созданном каталоге:

```
df -i /mnt/new_disk   # выводим сведения о количестве I-нодов
cd /mnt/new_disk/new_dir
echo "количество созданных файлов:"
ls -l | wc -l       # считаем количество файлов
```

11. Сохраните файл и закройте редактор. Настройте права на исполнение и запустите сценарий.

```
chmod +x ./test2.sh
./test2.sh
```

12. Начните работу с обнуляющим сценарием:

```
nano reset2.sh
```

13. Наберите шебанг и первые комментарии с описанием сценария:

```
#!/bin/sh
# сценарий удаления ФС
# Задание 2.1
```

14. Для возврата в исходное состояние нужно удалить созданные на новой ФС файлы и каталоги, размонтировать её, удалить файл самой системы и каталог в /mnt. Часть этих операций потребует прав суперпользователя:

```
cd /mnt/new_disk/          # переходим в ФС
rm -rf *                   # удаляем файлы
umount  /mnt/new_disk      # размонтируем ФС
rm /tmp/my_sys_2M.img      # удаляем файл my_sys_2M.img

rmdir /mnt/new_disk
```

Скрипт не отработал. Почему? Выполните изменения скрипта, добейтесь его работоспособности.

15. Сохраните файл и закройте редактор. Настройте права на исполнение и запустите сценарий и запустите сценарий с правами суперпользователя. После выполнения не забудьте завершить сеанс работы суперпользователя. .

```
chmod +x ./reset2.sh
su -
./reset2.sh
```

Задание 2.2

Выполните тестирование заполнения файловой системы по объему. При необходимости следуйте приведенным ниже пошаговым инструкциям.

1. Создайте сценарий **base2.2.sh**. В данном сценарии реализуйте:
   - создание файла **my_sys_2M.img** размером **2Мб** и файловой системы **ext2** в нем, аналогично тому, как это выполнялось в **Задании 2.1**.
   - создание каталога (точки монтирования) **/mnt/new_disk**,
   - монтирование в созданный каталог файловой системы.
   - вывод информации о файловых системах с помощью команды **df**
   - создание на примонтированной файловой системе рабочего каталога для размещения файлов /**mnt/new_disk/new_dir**
   - с помощью цикла **while** и инкрементально увеличивающегося счетчика осуществите создание в файловой системе файлов, аналогично тому, как описано в

Задании 2.1. Используйте команду **dd** для копирования файлов и устройство **/dev/zero** в качестве устройства-источника для команды **dd**. В опциях для команды **dd** также можно указать размер блока копируемого файла (bs) и количество копий (count).

- Например: **dd if=/dev/zero of=/mnt/new_disk/new_dir/<file> bs=10K count=1** **<file> —** имя создаваемого файла, подберите количество итераций цикла, значение **bs** и **count** таким образом чтобы достичь заполнение ФС по объему

2. Создайте сценарий **test2.2.sh**, осуществляющий проверку работы первого сценария, , аналогично тому, как это выполнялось в **Задании 2.1**. Сценарий должен выводить информацию по свободному месту и объему на файловой системе, сведения о общем и занятом количестве **inode** (индексных дескрипторов) и общее количество созданных циклом файлов.
3. Создайте сценарий **reset2.2.sh**, выполняющие аналогичные действия по “зачистке” результатов выполнения сценария **base2.2.sh**.

Лабораторная работа 3

Задание 3.1

1. Результат выполнения задания — это **три** сценария на языке командного интерпретатора.:
   первый (основной) выполняет задание;
   второй (тестовый) проверяет правильность работы основного;
   третий (обнуляющий) возвращает систему в исходное состояние так, чтобы можно было запустить процесс выполнения сценариев сначала.

2. Задача **основного** сценария задания:
   завести две локальные **групповые** учётные записи: **group1** и **group2**;
   создать три локальные **пользовательские** учётные записи: **user1**, **user2**, **user3**;
   включить пользователей в группы следующим образом:
   первый пользователь входит только в **первую** группу,
   второй пользователь — в **первую** и **вторую** группы,
   третий — только во **вторую**;
   создать в пределах ветки дерева каталогов /srv каталоги **dir1** и **dir2**;
   разграничить права следующим образом:
   подкаталог **dir1** доступен на запись только группе **group1**,
   dir2 — только группе **group2**.

3. Тестовый сценарий должен проверить соответствие прав на файлы, создаваемые в каталогах dir1 и dir2.

4. Для возврата к начальным условиям необходимо удалить пользователей и группы, а затем удалить созданные каталоги и файлы.

5. Для начала работы запустите текстовый редактор **nano**:
   nano base3.sh

6. Наберите шебанг и первые комментарии с описанием сценария:

```
#!/bin/sh
# сценарий создания групп пользователей
# Задание 3.1

# Дальше следуют команды создания групп и пользователей:
groupadd group1          # создаем группы
groupadd group2
```

```
adduser user1      # новые пользователи
adduser user2
adduser user3
gpasswd -a user1 group1  # добавляем пользователей в группы
gpasswd -a user2 group1
gpasswd -a user2 group2
gpasswd -a user3 group2
for i in $(seq 1 3)
do        # инфо о пользователях и группах
     echo "пользователь user$i"
     id user$i
done
mkdir /srv/dir1 /srv/dir2                # создаем каталоги
chgrp group1 /srv/dir1    # "отдаем" каталоги соотв. группам
chgrp group2 /srv/dir2
chmod 2775  /srv/dir1 /srv/dir2
# права на запись в каталоги
ls -la /srv
```

7. Сохраните файл и закройте редактор. Настройте права на исполнение и запустите сценарий и запустите сценарий с правами суперпользователя. После выполнения не забудьте завершить сеанс работы суперпользователя.
8. Разработайте тестовый сценарий (обратите внимание на различные варианты использования команды echo):

   `nano test3.sh`

```
#!/bin/sh
# сценарий проверки прав пользователей
# Задание 3.1
# USER2
su -l user2 -c '
cd /srv/dir1
touch file_u2d1
whoami && echo "создал(а) файл в каталоге" && pwd
cd /srv/dir2
touch file_u2d2
echo "`whoami` создал(а) файл в каталоге `pwd`"  '
# USER1
su -l user1 -c  '
# user1 - доступ к файлам в dir1
cd /srv/dir1/
```

```
whoami && echo "доступ к файлу в каталоге" && pwd
cp file_u2d1 file_u1d1
echo $?
# user1 - доступ к файлам в dir2
cd /srv/dir2/
echo "`whoami` создал(а) файл в каталоге `pwd`"
cp file_u2d2 file_u1d2
echo $?  '
```

9. Напишите аналогичную проверяющую секцию для пользователя **user3**.

   Затем сохраните файл и закройте редактор. Настройте права на исполнение и запустите сценарий.

10. Разработайте обнуляющий сценарий:

```
nano reset3.sh

#!/bin/sh
# сценарий удаления файлов, групп и пользователей
# Задание 3.1

# удаление каталогов и файлов
rm -rf /srv/dir1 /srv/dir2

# удаление групп и пользователей
userdel -r user1
userdel -r user2
userdel -r user3

groupdel group1
groupdel group2
```

11. Сохраните файл и закройте редактор. Настройте права на исполнение и запустите сценарий и запустите сценарий с правами суперпользователя. После выполнения не забудьте завершить сеанс работы суперпользователя.

Задание 3.2

В качестве самостоятельного задания разработайте три аналогичных сценария, при этом имена пользователей и названия групп должны задаваться как параметры сценария. Должна проводиться проверка количества передаваемых параметров. При необходимости ориентируйтесь на следующую пошаговую инструкцию.

1. Создайте файл базового сценария **base3.2.sh**. Базовый сценарий используйте такой же, как и в Задании 3.1 (то есть будут использоваться те же группы, пользователи и т.д.). **При запуске** сценария имена групп (`group1 и group2`) должны передаваться двумя первыми параметрами (**$1 и $2**), затем еще тремя параметрами должны передаваться имена пользователей (**$3, $4 и $5**), то есть скрипт должен запускаться в виде: **./base3.2.sh `group1 group2 user1 user2 user3`**

2. В начале сценария осуществите проверку количества переданных позиционных параметров с помощью условного оператора **if** и специальной переменной **$#** в которой содержится количество позиционных параметров, переданных в качестве аргументов при запуске сценария,
   пример конструкции условия:
   if [[$# -eq 5]]; then
   . . .
   fi
   При несоответствии количества переданных параметров условию следует осуществить вывод сообщения с информацией о правильном запуске и прекратить выполнение сценария.
3. Дальнейший код сценария пишется аналогично **Заданию 3.1**, но вместо имен групп и пользователей подставляются параметры, инициализированный при запуске скрипта, например:
   groupadd $1
   adduser $3
   gpasswd -a $3 $1
   chgrp $1 /srv/dir1
   и т. д.
4. Аналогично **Заданию 3.1** выполните создание проверочного сценария **test3.2.sh**. При этом запуск проверочного сценария должен осуществляться аналогично запуску основного сценария **base3.2.sh**, то есть должны передаваться параметры в качестве аргументов сценария:
   ./test3-2.sh group1 group2 user1 user2 user3
   Также должна осуществляться проверка количества передаваемых параметров с помощью условия **if** и специальной переменной **$#**
5. В проверочном скрипте также используйте подстановку параметров, вместо имен пользователей и групп. Сама логика сценария не меняется (код аналогичен Заданию 3.1). Не забудте добавить проверяющую секцию для пользователя **user3.** См. **Задание 3.1 п.9**
6. Сценарий очистки **reset3.2.sh** подготовьте также аналогично **Заданию 3.1**. При его запуске тоже передайте параметры и проверьте их количество с помощью условия **if**.

Лабораторная работа 4

Задание 4.1

1. В этом задании необходимо создать **четыре** сценария на языке командного интерпретатора.:
   стартовый (подготовительный) задает рабочую среду;
   первый (основной) выполняет задание;
   второй (тестовый) проверяет правильность работы основного;
   третий (обнуляющий) возвращает систему в исходное состояние так, чтобы можно было запустить процесс выполнения сценариев сначала.

2. Задача **стартового** сценария задания:
   запустить в фоновом режиме 5 процессов;
   3 из которых являются **целевыми**, а 2 — **дополнительными**.
   Для примеров фоновых процессов можно выбрать ping ya.ru и ping yandex.ru.

3. **Основной** сценарий должен прекратить выполнение только **целевых** процессов.

4. **Тестовый** сценарий должен сообщать сколько целевых и дополнительных процессов выполняется в данный момент.

5. Для возврата к начальным условиям необходимо прекратить выполнение всех запущенных процессов.

6. В этой лабораторной работе первым разработайте **тестовый** сценарий. Запустите текстовый редактор:
   ```
   nano test4.sh
   ```

7. Наберите шебанг и первые комментарии с описанием сценария:

   ```
   #!/bin/sh
   # сценарий мониторинга запущенных процессов
   # Задание 4.1

   echo "ping по адресу ya.ru"  #количество основных процессов
   pgrep -f ya.ru| wc -l

   echo "ping по адресу yandex.ru"   # количество доп. процессов
   pgrep -f yandex.ru| wc -l
   ```

8. Сохраните файл и закройте редактор. Настройте права на исполнение и запустите сценарий. При первом запуске в результате должны получиться нули.

9. Разработайте стартовый сценарий, :

```
nano start4.sh
```

```
#!/bin/sh
# сценарий запуска фоновых процессов
# Задание 4.1

# запускаем 3 целевых процесса
for i in `seq 1 3`; do ping ya.ru >/dev/null & done

# запускаем 2 дополнительных процесса
for i in 1 2; do ping yandex.ru >/dev/null & done
```

10. Сохраните файл и закройте редактор. Настройте права на исполнение и запустите сценарий.
11. После этого снова запустите `test4.sh`, проверьте количество целевых и дополнительных процессов.
12. Основной сценарий должен «убить» процессы, связанные с ya.ru, и «пощадить» все остальные:

```
nano base4.sh
```

```
#!/bin/sh
# сценарий целевого прекращения процессов
# Задание 4.1

# выбираем и прекращаем процессы
for p in `pgrep -f ya.ru`
 do
 kill $p
 done
```

13. Сохраните файл и закройте редактор. Настройте права на исполнение и запустите сценарий. После чего запустите `test4.sh` и проверьте количество целевых и дополнительных процессов.
14. Очищающий сценарий прекращает все процессы, запущенные в ходе работы:

```
nano reset4.sh
```

```
#!/bin/sh
```

```
# Сценарий очистки для
# Задание 4.1

# прекращаем все процессы из Задания 4.1

for p in `pgrep -f ya.ru`
 do
 kill  $p
 done

for p in `pgrep -f yandex.ru`
 do
 kill  $p
 done
```

15. Сохраните файл и закройте редактор. Настройте права на исполнение и запустите сценарий. После чего запустите `test4.sh` и проверьте количество целевых и дополнительных процессов.

Задание 4.2

В качестве самостоятельного задания разработайте аналогичные сценарии, изменив их так, чтобы адреса сайтов (целевого и дополнительного) задавались как параметры при запуске сценария. Должна проводиться проверка количества передаваемых параметров. При необходимосте пользуйтесь следующей пошаговой инструкцией.

1) Подготовьте стартовый (подготовительный) сценарий **start4.2.sh**: аналогично упражнения в задании 3.2 осуществите проверку количества передаваемых параметров с помощью условия if и специальной переменной $#. В данном случае требуется инициализировать 2 позиционных параметра, передаваемых при запуске сценария (например ya.ru и yandex.ru). В теле сценария используйте подстановку параметров ($1 и $2) вместо имен целевых узлов.

2) В сценарии **test4.2.sh** также используйте проверку количества параметров и используйте их вместо имен узлов.

3) В сценарии **base4.2.sh** укажите условие, в котором допускается использование менее, чем двух параметров, передаваемых при запуске сценария: : **if [[$# -lt 2]]** и протестируйте, передав при запуске сценария только один параметр (например узел ya.ru)

4) В сценарии “очистки **reset4.2.sh** передаем при запуске сценария оба параметра и в теле сценария подставляем параметры вместо имен узлов.

Итоговая лабораторная работа

1. Скопируйте все файлы заданий лабораторных работ 1-4 в отдельный каталог.

2. В ходе этой лабораторной работы нужно создать три сценария на языке командного интерпретатора:

- первый должен переименовать все файлы в каталоге, добавив Вашу фамилию в начало имени каждого файла;
- второй - добавляет комментарий во вторую строчку каждого сценария (после шебанга), комментарий должен содержать Ваше признание в авторстве программы;
- третий, как обычно, уничтожает работу основных сценариев.

Можно дополнительно написать четвертый сценарий, объединяющий работу первых двух.