

Лабораторная работа №5 по операционной системе

Имя: Чу минь Тиеп

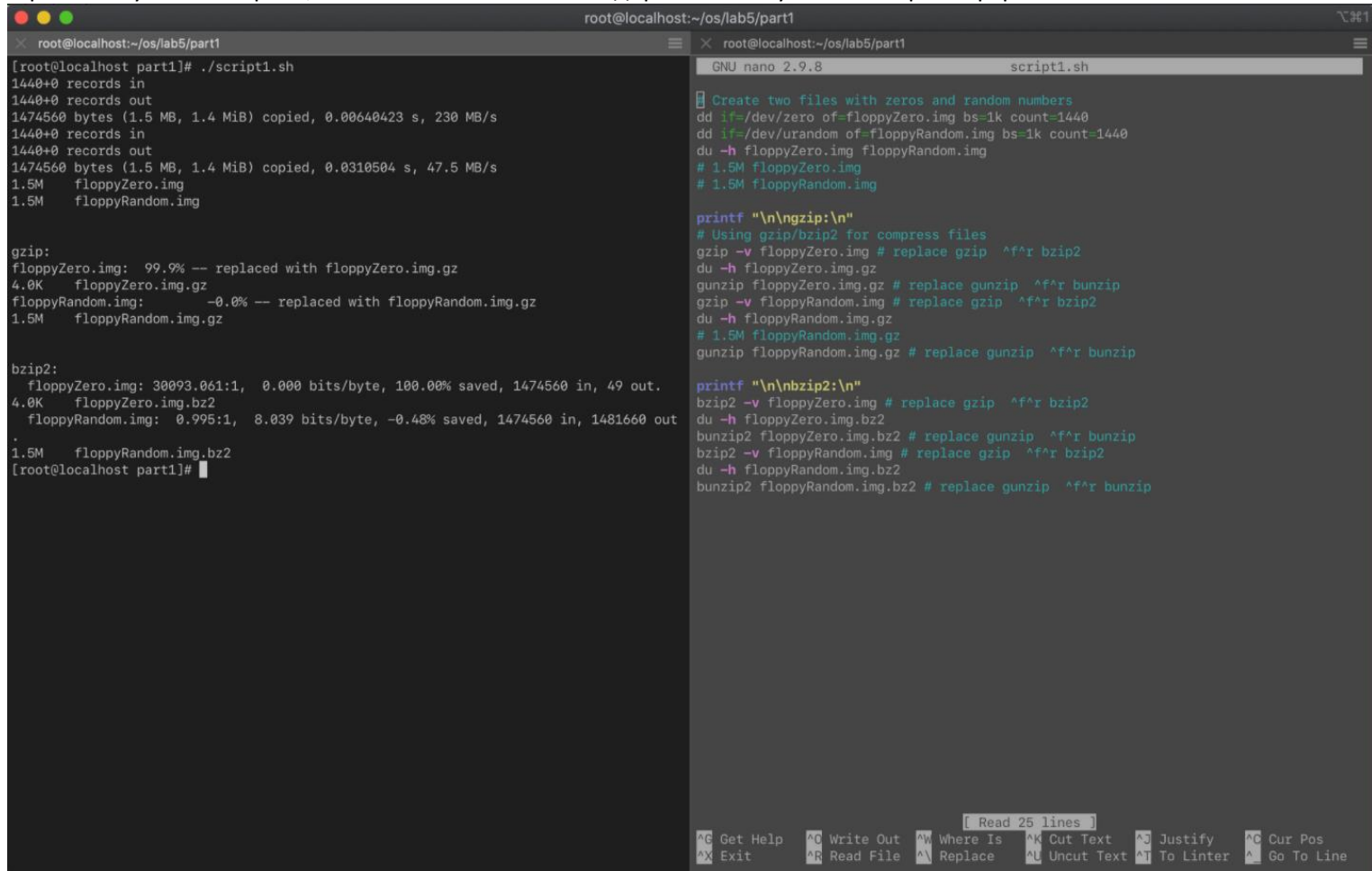
Группа: K33401

Преподаватель: Ватьян Александра Сергеевна

Часть 1. Сжатие данных

1. Объясните “странные” результаты сжатия файлов floppyZero и floppyRandom

- Эти алгоритмы отлично справляются с последовательностью идентичных символов, но практически бесполезны при отсутствии множества повторяющихся строк. Это связано с тем, что очень сложно найти повторяющиеся строки в случайной строке, это позволяет только закодировать их и уменьшить размер файла.



```
root@localhost:~/os/lab5/part1
[root@localhost part1]# ./script1.sh
1440+0 records in
1440+0 records out
1474560 bytes (1.5 MB, 1.4 MiB) copied, 0.00640423 s, 230 MB/s
1440+0 records in
1440+0 records out
1474560 bytes (1.5 MB, 1.4 MiB) copied, 0.0310504 s, 47.5 MB/s
1.5M floppyZero.img
1.5M floppyRandom.img

gzip:
floppyZero.img: 99.9% -- replaced with floppyZero.img.gz
4.0K floppyZero.img.gz
floppyRandom.img: -0.0% -- replaced with floppyRandom.img.gz
1.5M floppyRandom.img.gz

bzip2:
floppyZero.img: 30093.061:1, 0.000 bits/byte, 100.00% saved, 1474560 in, 49 out.
4.0K floppyZero.img.bz2
floppyRandom.img: 0.995:1, 8.039 bits/byte, -0.48% saved, 1474560 in, 1481660 out
1.5M floppyRandom.img.bz2
[root@localhost part1]#
```

```
GNU nano 2.9.8 script1.sh
# Create two files with zeros and random numbers
dd if=/dev/zero of=floppyZero.img bs=1k count=1440
dd if=/dev/urandom of=floppyRandom.img bs=1k count=1440
du -h floppyZero.img floppyRandom.img
# 1.5M floppyZero.img
# 1.5M floppyRandom.img

printf "\n\ngzip:\n"
# Using gzip/bzip2 for compress files
gzip -v floppyZero.img # replace gzip ^f^r bzip2
du -h floppyZero.img.gz
gunzip floppyZero.img.gz # replace gunzip ^f^r bunzip
gzip -v floppyRandom.img # replace gzip ^f^r bzip2
du -h floppyRandom.img.gz
# 1.5M floppyRandom.img.gz
gunzip floppyRandom.img.gz # replace gunzip ^f^r bunzip

printf "\n\nbzip2:\n"
bzip2 -v floppyZero.img # replace gzip ^f^r bzip2
du -h floppyZero.img.bz2
bunzip2 floppyZero.img.bz2 # replace gunzip ^f^r bunzip
bzip2 -v floppyRandom.img # replace gzip ^f^r bzip2
du -h floppyRandom.img.bz2
bunzip2 floppyRandom.img.bz2 # replace gunzip ^f^r bunzip
```

2. Отображает разницу в степени сжатия утилит с gzip

уровни сжатия 1, 6 и 9 (степень сжатия задается опцией -N, где N - число от 1

до 9, что указывает степень сжатия) и bzip2 с использованием примера сгенерированного файла

командой: **man man> heManFile**

The image shows a terminal window with two panes. The left pane shows the execution of a script named `script2.sh`. The output indicates that a file named `heManFile` is being replaced with its gzipped version (`heManFile.gz`) three times, each time showing a reduction in size (61.5%, 66.6%, and 66.7%). The final output shows the file size after compression: `heManFile: 3.339:1, 2.396 bits/byte, 70.05% saved, 37361 in, 11188 out.` The right pane shows the contents of the `script2.sh` script in the `nano` editor. The script defines a variable `heManFile` and then performs a series of operations: `gzip -1 -v $Name`, `du -h $Name.gz`, `gunzip $Name.gz`, `gzip -6 -v $Name`, `du -h $Name.gz`, `gunzip $Name.gz`, `gzip -9 -v $Name`, `du -h $Name.gz`, `gunzip $Name.gz`, `bzip2 -v $Name`, `du -h $Name.bz2`, and `bunzip2 $Name.bz2`. The terminal window title is `root@localhost:~/os/lab5/part1`.

```
root@localhost:~/os/lab5/part1
[root@localhost part1]# ./script2.sh
heManFile: 61.5% -- replaced with heManFile.gz
16K  heManFile.gz
heManFile: 66.6% -- replaced with heManFile.gz
16K  heManFile.gz
heManFile: 66.7% -- replaced with heManFile.gz
16K  heManFile.gz
heManFile: 3.339:1, 2.396 bits/byte, 70.05% saved, 37361 in, 11188 out.
12K  heManFile.bz2
[root@localhost part1]#
```

```
GNU nano 2.9.8 script2.sh

Name="heManFile"

gzip -1 -v $Name
du -h $Name.gz
gunzip $Name.gz

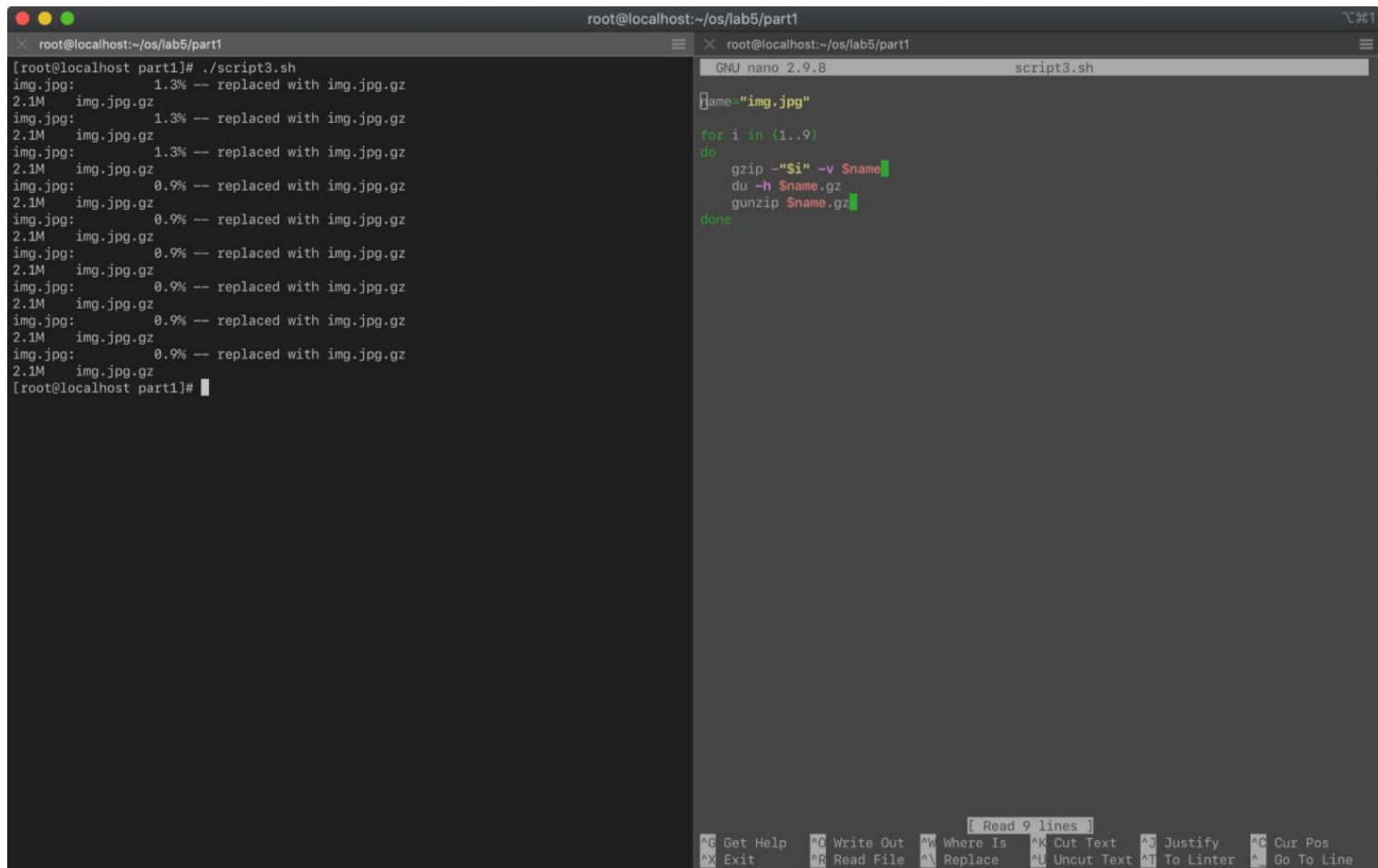
gzip -6 -v $Name
du -h $Name.gz
gunzip $Name.gz

gzip -9 -v $Name
du -h $Name.gz
gunzip $Name.gz

bzip2 -v $Name
du -h $Name.bz2
bunzip2 $Name.bz2
```

3. Попробуйте сжать картинку в формате png (jpeg) утилитой `gzip`, используя различные уровни сжатия от 1 до 9. На сколько процентов от исходного размера был сжат файл? Сильно ли отличается степень сжатия между уровнями? Почему?

-Файл сжимается довольно сильно, во всех случаях около 99% нового размера. Это неэффективное сжатие происходит из-за того, что форматы JPG / PNG уже являются сжатыми файлами и практически не содержат избыточной информации.



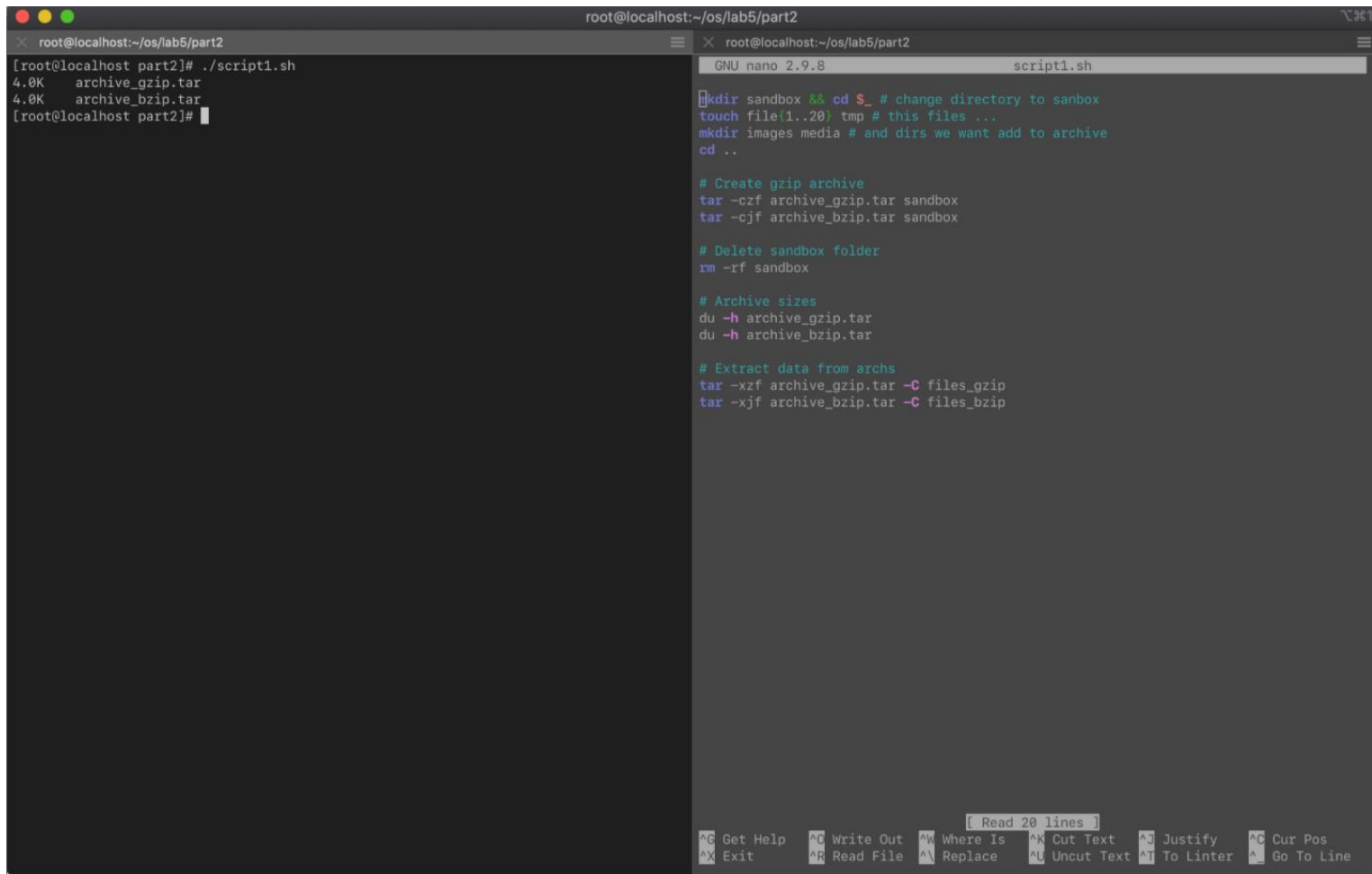
The screenshot shows a terminal window with two panes. The left pane displays the output of a script named `script3.sh` being executed. The output shows a loop where `img.jpg` is repeatedly replaced with `img.jpg.gz`, with file sizes (2.1M) and percentages (1.3% and 0.9%) indicating the progress. The right pane shows the `script3.sh` file being edited in the `nano` editor. The script content is as follows:

```
name="img.jpg"
for i in {1..9}
do
  gzip -"$i" -v $name
  du -h $name.gz
  gunzip $name.gz
done
```

The terminal window title is `root@localhost:~/os/lab5/part1`. The nano editor status bar at the bottom indicates "Read 9 lines" and lists various keyboard shortcuts.

Часть 2. Архивирование данных

Как уже упоминалось, `tar` не умеет самостоятельно сжимать данные, но вы можете попросить `tar` использовать одну из утилит для сжатия данных, например `gzip` / `bzip2`. Замените параметры при создании / распаковке архива `-cf` / `-xf` на `-czf` / `-xzf` (для использования `gzip`) или на `-cjz` / `-xjf` (для использования `bzip2`), затем сравните размеры архивный файл со сжатием данных и без него



The screenshot shows a terminal window with two panes. The left pane shows the execution of a script named `script1.sh`. The output indicates that two archives, `archive_gzip.tar` and `archive_bzip.tar`, each 4.0K in size, were created. The right pane shows the contents of `script1.sh` in the nano editor. The script performs the following actions:

- Creates a `sandbox` directory and changes to it.
- Creates two files, `file{1..20}`, in the `tmp` directory.
- Creates two directories, `images` and `media`.
- Creates a gzip archive `archive_gzip.tar` and a bzip archive `archive_bzip.tar` from the `sandbox` directory.
- Deletes the `sandbox` folder.
- Displays the sizes of the created archives using `du -h`.
- Extracts the data from the archives into `files_gzip` and `files_bzip` directories.

```
GNU nano 2.9.8 script1.sh

mkdir sandbox && cd $_ # change directory to sandbox
touch file{1..20} tmp # this files ...
mkdir images media # and dirs we want add to archive
cd ..

# Create gzip archive
tar -czf archive_gzip.tar sandbox
tar -cjf archive_bzip.tar sandbox

# Delete sandbox folder
rm -rf sandbox

# Archive sizes
du -h archive_gzip.tar
du -h archive_bzip.tar

# Extract data from archs
tar -xzf archive_gzip.tar -C files_gzip
tar -xjf archive_bzip.tar -C files_bzip
```

Часть 3. Создание различных видов бэкапов

Напишите сценарий bash для создания инкрементных резервных копий. Используйте те же команды, что и для резервного копирования дельты.

```
# Schema:
# to_backup/ - files to backup
# backup/ - backups
# restored/ - restored files

# Create files for backup
mkdir to_backup
touch to_backup/file{1..3}

# Create folder for backup
mkdir backup 2> /dev/null
cd backup

# Create backup
tar -cpvzf full_backup.tar.gz.0 -g backup.snap ../to_backup
cp backup.snap backup.snap.1
touch ../to_backup/file4
tar -cpvzf diff_backup.tar.gz.1 -g backup.snap.1 ../to_backup
cat backup.snap.1

for i in {1..3}
do
    cp "backup.snap.$i" "backup.snap.$((i+1))"
    touch "../to_backup/file$((i+4))"
    tar -cpvzf "diff_backup.tar.gz.$((i+1))" -g "backup.snap.$((i+1))" ../to_backup
    cat "backup.snap.$((i+1))"
done

cd ..
mkdir restored
tar -xvf backup/full_backup.tar.gz.0 -G -C restored

for i in {1..4}
do
    tar -xvf backup/diff_backup.tar.gz.$i -G -C restored
done
```

```
root@localhost:~/os/lab5/part3
[root@localhost part3]# ./backup_diff.sh
tar: ../to_backup: Directory is new
tar: Removing leading `../' from member names
../to_backup/
../to_backup/file1
../to_backup/file2
../to_backup/file3
tar: Removing leading `../' from member names
../to_backup/
../to_backup/file4
GNU tar-1.30-2
1606776435396006405016067764353925926996476834111624../to_backupNfile1Nfile2Nfile3Yfi
le4tar: Removing leading `../' from member names
../to_backup/
../to_backup/file5
GNU tar-1.30-2
1606776435404610264016067764354015927006476834111624../to_backupNfile1Nfile2Nfile3Nfi
le4Yfile5tar: Removing leading `../' from member names
../to_backup/
../to_backup/file6
GNU tar-1.30-2
160677643541610179016067764354135927006476834111624../to_backupNfile1Nfile2Nfile3Nfi
le4Nfile5Yfile6tar: Removing leading `../' from member names
../to_backup/
../to_backup/file7
GNU tar-1.30-2
1606776435424355064016067764354215927016476834111624../to_backupNfile1Nfile2Nfile3Nfi
le4Nfile5Nfile6Yfile7to_backup/
to_backup/file1
to_backup/file2
to_backup/file3
to_backup/
to_backup/file4
to_backup/
to_backup/file5
to_backup/
to_backup/file6
to_backup/
to_backup/file7
[root@localhost part3]# ls
backup backup_diff.sh restored to_backup
[root@localhost part3]# ls restored
to_backup
[root@localhost part3]# ls restored/to_backup/
file1 file2 file3 file4 file5 file6 file7
[root@localhost part3]# ls to_backup/
file1 file2 file3 file4 file5 file6 file7
[root@localhost part3]#
```

```
GNU nano 2.9.8 backup_diff.sh Modified
# Schema:
# to_backup/ - files to backup
# backup/ - backups
# restored/ - restored files

# Create files for backup
mkdir to_backup
touch to_backup/file{1..3}

# Create folder for backup
mkdir backup 2> /dev/null
cd backup

# Create backup
tar -cpvzf full_backup.tar.gz.0 -g backup.snap ../to_backup
cp backup.snap backup.snap.1
touch ../to_backup/file4
tar -cpvzf diff_backup.tar.gz.1 -g backup.snap.1 ../to_backup
cat backup.snap.1

for i in {1..3}
do
    cp "backup.snap.$i" "backup.snap.${(i+1)}"
    touch "../to_backup/file${(i+4)}"
    tar -cpvzf "diff_backup.tar.gz.${(i+1)}" -g "backup.snap.${(i+1)}" ../to_backup
    cat "backup.snap.${(i+1)}"
done

cd ..
mkdir restored
tar -xvf backup/full_backup.tar.gz.0 -G -C restored

for i in {1..4}
do
    tar -xvf backup/diff_backup.tar.gz.$i -G -C restored
done

^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos
^X Exit ^R Read File ^\ Replace ^U Uncut Text ^T To Linter ^_ Go To Line
```

Часть 4. Автоматизация выполнения заданий

1. Для выполнения задач сразу используйте команду `at`. С помощью этой команды запланируйте одноразовое полное резервное копирование.
-tar -cpvzf full_backup.tar.gz for_backup | at 20:30
2. Используя cron и сценарий из предыдущей части, запланируйте выполнение разностных или инкрементных бэкапов

```
EDITOR=nano
crontab -e

# In file: m h D M WD
00 12 1 * * /root/os/lab5/part3/backup_diff.sh
```

