Lab 2: The filesystem, command line, and file manipulation

Ahmed Baha Eddine Alimi

Assignment Report

SD-01

I. Filesystem:

1. How many inodes are in use on your system?

My system is using: 270541 inodes

I checked through `df -i` command:

```
allimi@Ubuntu:~/Desktop$ df -i
Filesystem      Inodes  IUsed   IFree IUse% Mounted on
tmpfs           499177    954  498223    1% /run
/dev/sda3      1605632 270541 1335091   17% /
tmpfs           499177      1  499176    1% /dev/shm
tmpfs           499177      4  499173    1% /run/lock
efivarfs             0      0       0     - /sys/firmware/efi/efivars
/dev/sda2            0      0       0     - /boot/efi
tmpfs            99835    141   99694    1% /run/user/1000
```

2. What is the filesystem type of the EFI partition?

My EFI partition filesystem type is `vfat`

```
allimi@Ubuntu:~/Desktop$ sudo blkid | grep EFI
[sudo] password for allimi:
/dev/sda2: UUID="93AB-129B" BLOCK_SIZE="512" TYPE="vfat" PARTLABEL="EFI System Partition" PARTUUID="0db50176-06de-4009-be2b-29a247d59050"
allimi@Ubuntu:~/Desktop$
```

3. What device is mounted at your root / directory? Show proof.

My root device is `/dev/sda3`

```
allimi@Ubuntu:~/Desktop$ df /
Filesystem     1K-blocks     Used Available Use% Mounted on
/dev/sda3       25106692 14881152   8924856  63% /
```

4.What is your partition UUID?

```
allimi@Ubuntu:~$ blkid
/dev/sda3: UUID="cb7e8d22-d97f-4733-b8b5-b1c553281609" BLOCK_SIZE="4096" TYPE="ext4" PARTUUID="c5c55913-dd0c-4531-8d1e-7dc0e01cbf0e"
/dev/loop1: TYPE="squashfs"
/dev/loop8: TYPE="squashfs"
/dev/loop6: TYPE="squashfs"
/dev/loop4: TYPE="squashfs"
/dev/loop2: TYPE="squashfs"
/dev/loop0: TYPE="squashfs"
/dev/loop9: TYPE="squashfs"
/dev/loop7: TYPE="squashfs"
/dev/sda2: UUID="93AB-129B" BLOCK_SIZE="512" TYPE="vfat" PARTLABEL="EFI System Partition" PARTUUID="0db50176-06de-4009-be2b-29a247d59050"
/dev/loop5: TYPE="squashfs"
/dev/loop3: TYPE="squashfs"
/dev/loop10: TYPE="squashfs"
```

`/dev/sda3: UUID=”cb7e8d22-d97f-4733-b8b5-b1c553281609”`

`/dev/sda2: UUID="93AB-129B"`

5.What is the function of /dev/zero?

`/dev/zero` is a special file that provides an endless stream of null (zero) bytes (`\0`), It is often used for creating empty files with `dd`, initializing memory in low-level programming and overwriting disks for secure deletion

II. Command line and file manipulation:

1. Explain the role of the Pipe `|` in this command `cat /etc/apt/sources.list | less`.

The pipe (`|`) redirects the output of `cat /etc/apt/sources.list into less`, which allows for scrolling through the file interactively.

2. What does section 5 in man mean? And how can you find it?

man 5 contains documentation for file formats and configuration files.
We can find with this command template: `man 5 <command>`

3. What is the full file path of `ls` on your machine? How did you find it?

The full path of ls in my machine is `/usr/bin/ls`

```
allimi@Ubuntu:~/Desktop$ which ls
/usr/bin/ls
```

I used `which ls` command

4.Show two ways of renaming a file `test_file.tot` to `test_file.txt`.

We can use the `mv` command:

```
allimi@Ubuntu:~/Desktop$ touch test_file.tot
allimi@Ubuntu:~/Desktop$ mv test_file.tot test_file.txt
```

Or through the `rename` command:

```
allimi@Ubuntu:~/Desktop$ rename 's/\.tot$/.txt/' test_file.tot
allimi@Ubuntu:~/Desktop$
```

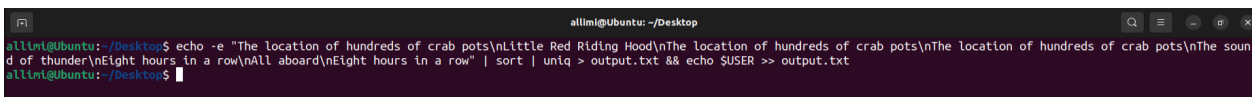
5.Create a compound command that does the following to a given string:

- sort the given string
- find only the unique lines without duplication
- save the sorted unique lines to a file
- append the username of the currently logged in user to the end of the file.
- The given string is below:

```
The location of hundreds of crab pots\nLittle Red Riding Hood\nThe location of hundreds of crab pots\nThe location of hundreds of crab pots\nThe sound of thunder\nEight hours in a row\nAll aboard\nEight hours in a row
```

I used:

```
echo -e "The location of hundreds of crab pots\nLittle Red Riding Hood\nThe location of hundreds of crab pots\nThe location of hundreds of crab pots\nThe sound of thunder\nEight hours in a row\nAll aboard\nEight hours in a row" | sort | uniq > output.txt && echo $USER >> output.txt
```



`echo -e` **prints the string with new lines.**

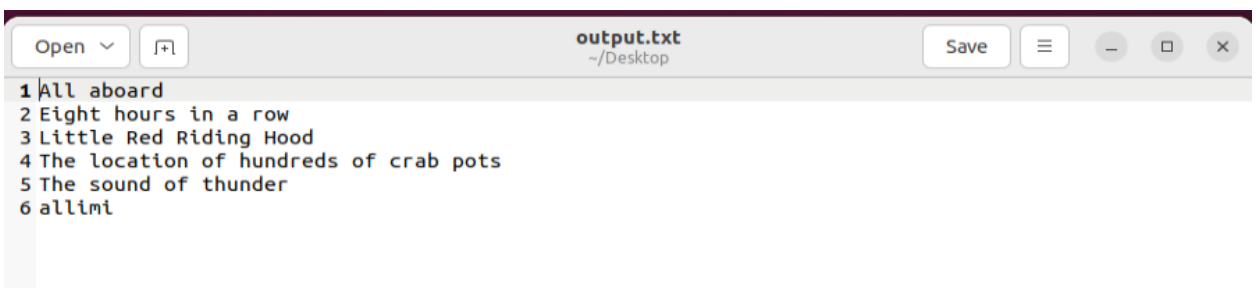
`sort` **sorts the lines.**

`uniq` **removes duplicates.**

`>` **writes the sorted unique lines to output.txt.**

`&& echo $USER >> output.txt` **appends the current username.**

Output:



6.What can you do to discard the output from the command ping `127.0.0.1`? You should also discard standard error. Show how you achieve this.

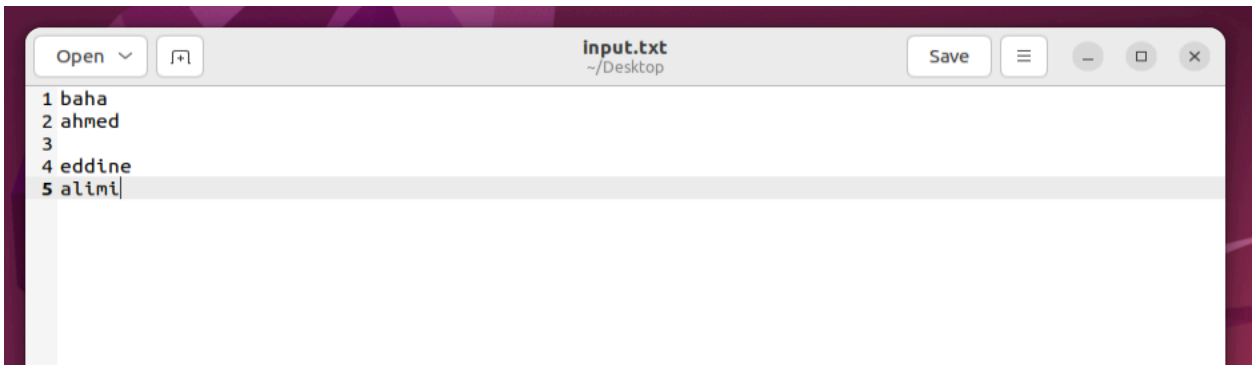
```
ping 127.0.0.1 > /dev/null 2>&1
```

```
allimi@Ubuntu:~/Desktop$ ping 127.0.0.1 > /dev/null 2>&1
```

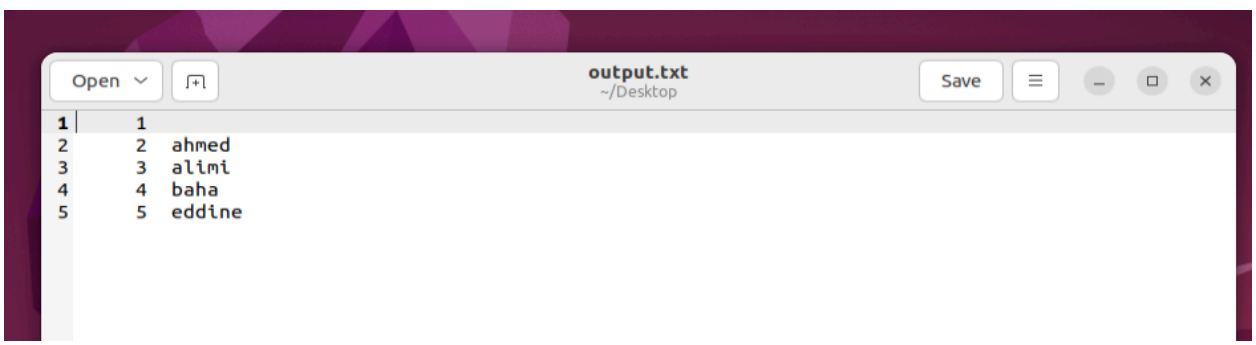
`> /dev/null` **discards standard output.**
`2>&1` **redirects standard error to standard output (which is already discarded).**

7.Show how you can sort input, append line numbers, and save the sorted result to a file. Add line numbers to empty lines too.

```
allimi@Ubuntu:~/Desktop$ sort input.txt | nl -ba > output.txt
allimi@Ubuntu:~/Desktop$
```



Output:



`sort input.txt` **sorts the lines.**
`nl -ba` **numbers all lines (including empty ones).**
`> output.txt` **saves the output.**

8.Create the directory `/home/$USER/testdir`. Write out as much as possible ways to go from `/usr/share` folder to `/home/$USER/testdir`.

Making the directory:

```
allimi@Ubuntu:~/Desktop$ mkdir -p /home/$USER/testdir
```

going from `/usr/share` folder to `/home/$USER/testdir`:

Using Absolute Path with `cd`:

`cd /home/$USER/testdir`

Using Relative Paths with `cd`:

`cd ../../home/$USER/testdir`

Using `cd` with ~:

`cd ~/testdir`

Using `pushd` and `popd` (Stack Navigation):

```
allimi@Ubuntu:~$ pushd /home/$USER/testdir
~/testdir ~
allimi@Ubuntu:~/testdir$ popd
```

9.Write a pipe that will result with a unique list of commands`/shell` from `/etc/passwd` file (last column of it)

`cut -d: -f7` **extracts the last column (shells).**

`sort -u` **sorts and removes duplicates.**

```
allimi@Ubuntu:~/Desktop$ cut -d: -f7 /etc/passwd | sort -u
/bin/bash
/bin/false
/bin/sync
/usr/sbin/nologin
allimi@Ubuntu:~/Desktop$
```

III. Bonus Questions:

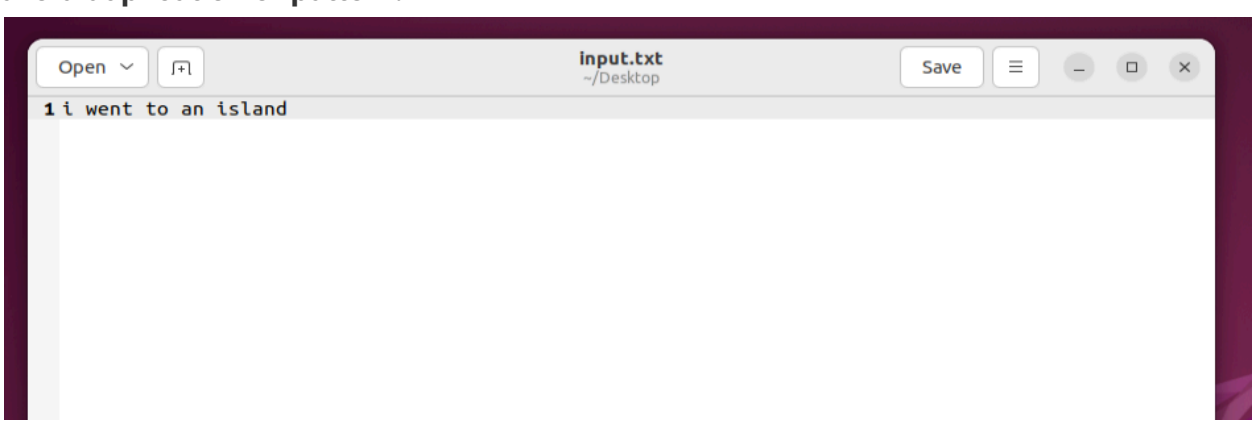
1.Find all man pages that contain word malloc. The result should be just a list of files

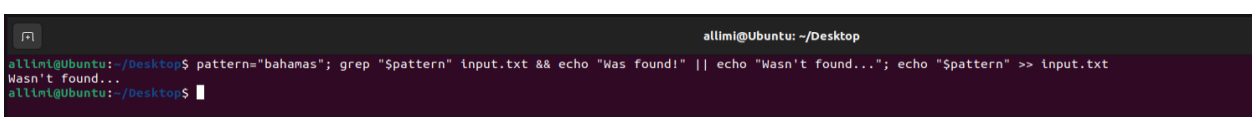
```
allimi@Ubuntu:~/Desktop$ apropos . | grep malloc | awk '{print $1}'
__after_morecore_hook
__free_hook
__malloc_hook
__malloc_initialize_hook
__memalign_hook
__realloc_hook
malloc
malloc_get_state
malloc_hook
malloc_info
malloc_set_state
malloc_stats
malloc_trim
malloc_usable_size
mtrace
mtrace
muntrace
```

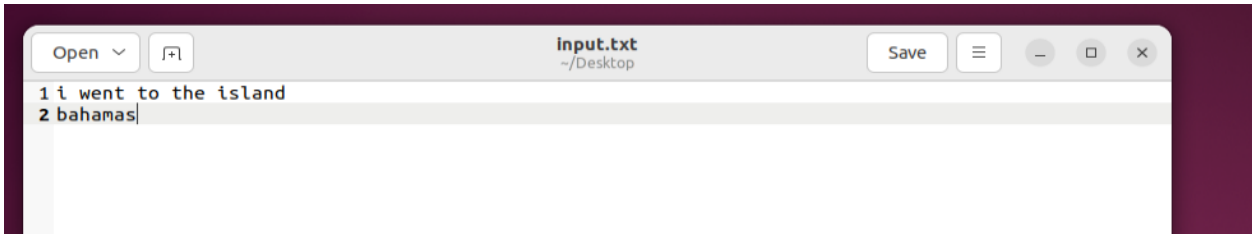
- `apropos .` searches all man pages.
- `grep malloc` filters the results to only include lines containing the word "malloc".
- `awk '{print $1}'` extracts just the man page names.

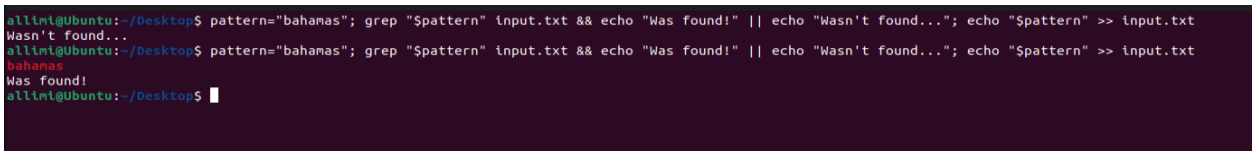
2.Write a one-liner that will result with a message "Was found!" if grep found an occurence of a pattern, and "Wasn't found..." if grep does not find the pattern. Add the desired pattern as the last line of any file.

Your one-liner should look like: `grep <pattern> filename` . . . . Use variables to avoid duplication of pattern.









- `pattern="bahamas"` sets the pattern to search for.
- `grep "$pattern" input.txt` searches for the pattern in the file.
- `&& echo "Was found!"` prints "Was found!" if the pattern is found.
- `|| echo "Wasn't found…"` prints "Wasn't found…" if the pattern is not found.
- `echo "$pattern" >> input.txt` appends the pattern to the file as the last line.