Activity 3 : Linux

Group No : 27

Group Member :

1. Waranthorn Chansawang
2. Anik Romyanon
3. Ittipat Yodprasit

Part 4 Swap

4.1 What is swap ?

RAM is a luxury not all of us have in plenty. Thus, swap, a.k.a pagefile, is invented to provide the use of hard drive storage as a temporary extended memory, of course, it is much much much slower, but for applications that are not in use right now it is generally good enough. At least, it prevents the system from running out of memory.

4.2 Finding the amount of available memory

* DO THIS *

Try the command

```
free
```

Also, try

```
free -m
```

* ANSWER THIS * How can you find out how much memory is available? What are the units of the value shown as the results of the command above?

```
Check the right most part (“available”)
```

```
for free the unit is bytes
for free -m the unit is kilobytes
```

4.3 Life without swap

In this part, we will use Python to take up as much available memory as possible (until it crashes!) by assigning a value that is bigger than the available memory to a variable in Python.

* DO THIS * With the help of `bytearray(size_in_bytes), try the following command.`

```
>>> a = bytearray(600000*1024)
```

If things go well, the variable will be created and take up the memory as allocated by the input argument of `bytearray()`.

Try hitting ctrl+z to suspend the "foreground" process, which is Python. This will bring us back to the shell so that we can execute the free command again (with Python still running in the background).

```
[1]+  Stopped                 python
```

Remember the [1] well this is the process id, we need this number to resume back to the process.

Observe the result of `free` again. * ANSWER THIS * Do you see any evidence that the variable '`a`' in the Python process takes a large amount of the memory available previously? Explain.

```
The memory in available is reduced by a large amount (previously
from 600000 down to 100000)f
```

* DO THIS * Now, let's go back to the Python process we have just suspended, with the command **fg**:

```
fg [process_id]
```

* DO THIS * Next, we want to try to find the biggest value we can allocate.

- Use `del a` to delete the variable (free up the memory).
- Assign a new value to the variable 'a' of `bytearray()`.

Repeat the trial process until you encounter a `MemoryError` or `Killed` which means the OS cannot allocate the memory for you.

* ANSWER THIS * What is the largest memory size you can allocate with **bytearray** ?

```
Around 700000 kilobytes
```

4.4 Create and enable the swap file

Swap in Linux can help us allocate more memory than our RAM can provide. This is done by creating a swapfile, or a swap partition, but most of the time a swapfile is more preferable.

* DO THIS * In this part, we will make an extra 1GB available to the system with a swapfile.

To create a swapfile at /, and enable it, we can do this:

```
# create a file with 1 GB of size
sudo fallocate -l 1G /swapfile
# set the file permission to be root-only
sudo chmod 600 /swapfile
# make it a swapfile
sudo mkswap /swapfile
# register to the system
sudo swapon /swapfile
# check for yourself
free
```

Now, we have 1 GB of swap, we should now be able to allocate more memory.

4.5 Life with swap

* ANSWER THIS * After having the swapfile created earlier, what is the largest memory size you can allocate with **bytearray** right row ?

```
Around 1200000 kilobytes
```

4.6 Make swap permanent

Even though our swapfile is active, it will not after a system restarts. To make it permanent we have to tell the system to “mount” this file as swap every time the system starts.

```
sudo nano /etc/fstab
```

Add the following line to the end of file:

```
/swapfile none swap sw 0 0
```

Generally, **fstab** controls what will be mounted after the system starts.

Now, let’s restart the system using `sudo reboot`.

You will be disconnected, but the reboot process should not take long, wait a minute, and try reconnecting.

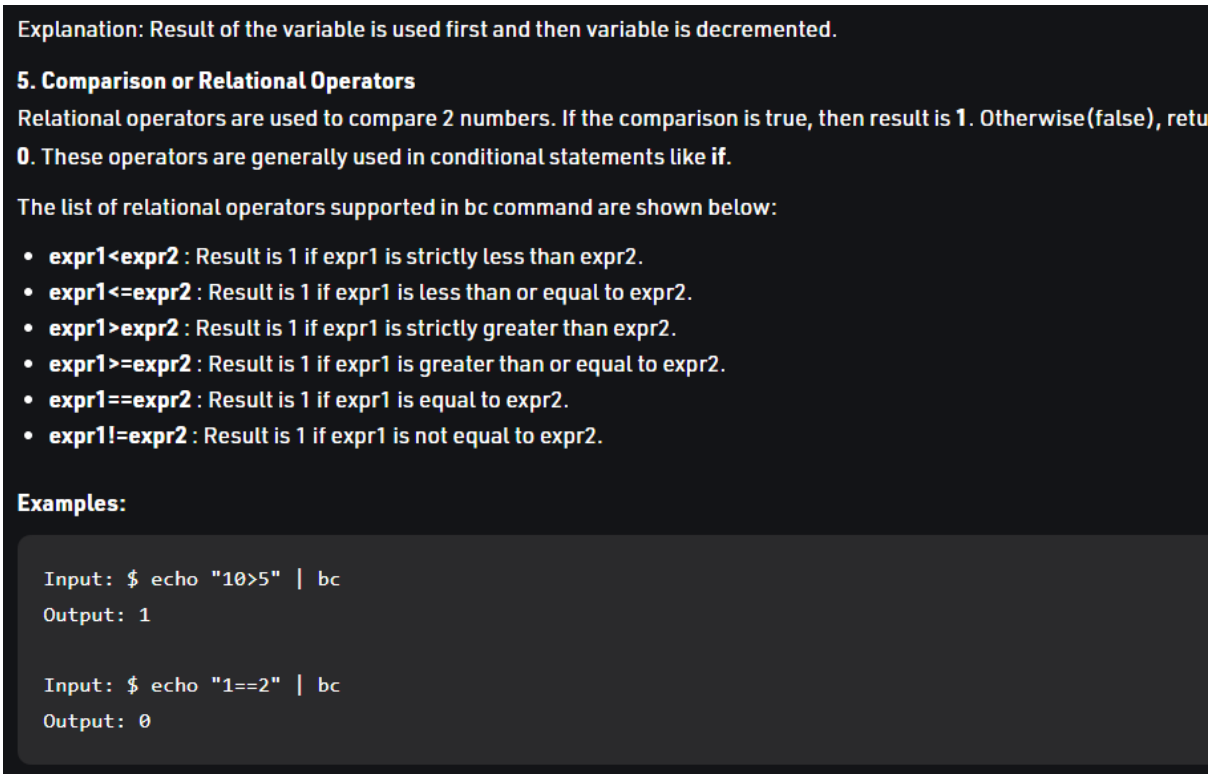
See **free** for yourself that the swap is automatically mounted after restart.

4.7 * SHOW THIS * Show your result to a teaching staff.

```
film@g-27:~$ free
               total        used        free      shared  buff/cache   available
Mem:           994832      131416      578732         956      284684      717084
Swap:         1048572           0     1048572
```

https://www.geeksforgeeks.org/bc-command-linux-examples/

https://linuxize.com/post/bash-if-else-statement/



Explanation: Result of the variable is used first and then variable is decremented.

5. Comparison or Relational Operators

Relational operators are used to compare 2 numbers. If the comparison is true, then result is **1**. Otherwise(false), retu
0. These operators are generally used in conditional statements like **if**.

The list of relational operators supported in bc command are shown below:

- **expr1<expr2** : Result is 1 if expr1 is strictly less than expr2.
- **expr1<=expr2** : Result is 1 if expr1 is less than or equal to expr2.
- **expr1>expr2** : Result is 1 if expr1 is strictly greater than expr2.
- **expr1>=expr2** : Result is 1 if expr1 is greater than or equal to expr2.
- **expr1==expr2** : Result is 1 if expr1 is equal to expr2.
- **expr1!=expr2** : Result is 1 if expr1 is not equal to expr2.

Examples:

```
Input: $ echo "10>5" | bc
Output: 1

Input: $ echo "1==2" | bc
Output: 0
```

A Chance to be “Outstanding”

Create a **bash** script called “Minicalcuator.sh” which is intended for calculating two numbers at the time the script is executed.

- There are 4 functions that this program can do:
    1. Addition
    2. Subtraction
    3. Multiplication
    4. Division
- Input of two numbers must be read to the program via keyboard and user has to **select** one of the four operations by its number(1-4) in order to perform specific calculation at a time.
- The output should be as following:

    Welcome to Mini Calculation program
    Enter the first number: 2

Enter the second number: 3
1) add
2) subtract
3) multiply
4) divide
Choose the operation: 1
2 + 3 = 5

- If user inputs an another choice other than choices in the list, the screen must show a **small flyable train animation** and go back to get the operation choice from user again unless the answer is correct.

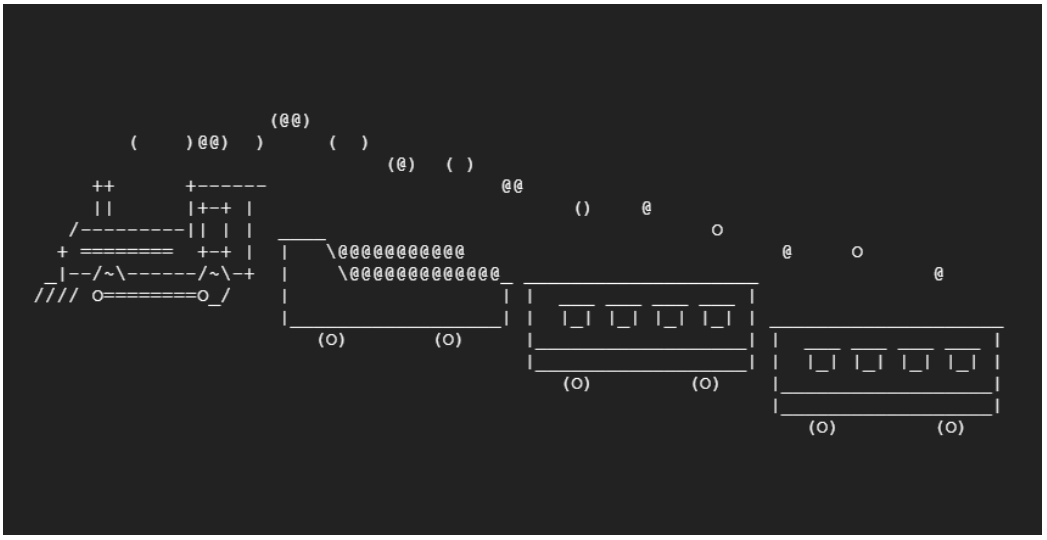
  *Note red numbers are from user input

Implement the job, explain how you did it, and show all relevant code:

The program will run as following

```
Welcome to Mini Calculation program
Enter the first number: 2
Enter the second number: 3
1) add
2) subtract
3) multiply
4) divide
Choose the operation: 1
2 + 3 = 5
```

If user does not input option 1,2,3 or 4, it will show this train is flying from bottom right to upper left corner and go back to get option number again.



Hints

- Google “bash bc” might ease your life.
- If you get **Permission denied** error, try to change permission of your file or your directory.