1) Learn shell commands

    chmod, chown, set,  cut, time

2) change your prompt

3) add a new directory to your PATH, and display the new path.

4) Write and execute  shell script which uses the "date" command

    and displays whether it is morning, noon, afternoon, evening or night.

5) use of for loop and while loops

6) reading from termina

Q1. Using awk find the number of users using /bin/sh and /bin/bash

cat /etc/passwd

gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/bin/sh

libuuid:x:100:101::/var/lib/libuuid:/bin/sh

syslog:x:101:102::/home/syslog:/bin/bash

hplip:x:103:7:HPLIP system user,,,:/var/run/hplip:/bin/false

saned:x:110:116::/home/saned:/bin/bash

pulse:x:111:117:PulseAudio daemon,,,:/var/run/pulse:/bin/ksh

gdm:x:112:119:Gnome Display Manager:/var/lib/gdm:/bin/bash

Q2. With respect to the above file, what the following awk commands do?

     What is the meaning of NF?

$ awk -F ':' '{ total += NF }; END { print total }' /etc/passwd

$ awk -F ':' '$3 > maxuid { maxuid=$3; maxline=$0 }; END { print maxuid, maxline }' /etc/passwd

$ awk 'NR % 2 == 0' /etc/passwd

$awk -F ':' '$3==$4' passwd.txt

Q3. What does this awk script do?

#!/bin/awk -f

BEGIN {

FS=":";

}

{

if ( $2 == "" ) {

print $1 ": no password!";

}

}

 (to be continued with more examples)

<http://kirste.userpage.fu-berlin.de/chemnet/use/info/gawk/gawk_toc.html>

Pl have a look at this.

Try to understand all regular expressions like

^p

p$

p?

p\*

p+   etc

Given a file consisting of lines, use 'awk' to find the frequency of words

in the file.

1) Go through attached slides.

2) Run the programs involving fork(), wait() and exec()

3) See how use of the above three functions can make one to

 write his/her own shell

4) understand command line arguments in the process.

**Parent and child processes are usually independent and they execute in different address space. However, there is a need to communicate between them to perform many useful activities. pipe() is a system call to achieve this. This system call is used for one-way communication.**

**1) Read the details of pipe() system call from the link**

[**https://www.tutorialspoint.com/inter\_process\_communication/inter\_process\_communication\_pipes.htm**](https://www.tutorialspoint.com/inter_process_communication/inter_process_communication_pipes.htm)

**2) understand the examples given in this site.**

**3) use pipe(s) to solve the following problem.**

**Write a Program for validation of a Sorting Program executed by child processes.**

**The parent process will give an unsorted array to 2 child processes. One child will**

**run correct version of the sorting program and the other child will run an incorrect**

**version. Then, the results will be communicated to the parent process which will**

**validate the results with the sorted output.**

[**https://www.tutorialspoint.com/inter\_process\_communication/inter\_process\_communication\_named\_pipes.htm**](https://www.tutorialspoint.com/inter_process_communication/inter_process_communication_named_pipes.htm)

[**https://www.geeksforgeeks.org/named-pipe-fifo-example-c-program/**](https://www.geeksforgeeks.org/named-pipe-fifo-example-c-program/)

**1) use named-pipe to solve the following problem.**

**Write a Program for validation of a Sorting Program executed by child processes.**

**The parent process will give an unsorted array to 2 child processes. One child will**

**run correct version of the sorting program and the other child will run an incorrect**

**version. Then, the results will be communicated to the parent process which will**

**validate the results with the sorted output.**

**2) implement bounded buffer problem using shared memory and show that race condition occurs.**

**(both Producer and the Consumer processes  try to modify the same variable).**

   3**)  Read about threads in the followimg document, and execute and understand the program in Figure 26.2**

<http://pages.cs.wisc.edu/~remzi/OSTEP/threads-intro.pdf>

<https://www.tutorialspoint.com/inter_process_communication/inter_process_communication_named_pipes.htm>

<https://www.tutorialspoint.com/inter_process_communication/inter_process_communication_shared_memory.htm>

**1) Show using two threads accessing a common variable that race-condition can occur.**

**2) Show how using multiple threads, we can efficiently compute multiplication**

**of two 3X3 matrices.**

**3)**[**https://www.tutorialspoint.com/java/java\_multithreading.htm**](https://www.tutorialspoint.com/java/java_multithreading.htm)

**Refer the above link and understand Java threads.**

**1) Implement a buddy memory allocator.**

**-- build an interface in which different processes will request for memory with**

**certain sizes**

**-- Let the buddy system allocate a buddy for each request**

**-- display the state of the buddy allocator: Which buddies are free and which**

**are allocated. The display should be a nice form such that it is easy to**

**understand.**

**Implementation of a page-based memory system.**

**Assume all pages are in Main memory.**

**Assume Page Table is stored in Main memory**

**Implement an interface in which processes demand pages.**

**Then Page table is constructed, and right links are established.**

**Then implement  virtual page to physical page translation**

**Implement the LRU replacement policy on top of the page based mM that you implemented last time.**

**1) You have already done an assignment on race condition.**

**Each of the two threads modify a common counter a large no of times.**

**At the end, it is observed that each run of the program gives different results due to**

**race conditions.**

**Use a semaphore to show that there is no race condition.**

**Please refer to**

[**http://pages.cs.wisc.edu/~remzi/OSTEP/threads-sema.pdf**](http://pages.cs.wisc.edu/~remzi/OSTEP/threads-sema.pdf)

**for use of semaphores.**

1**) Producer consumer problem using threads**

**2) Producer consumer problem using shared memory**

**Reader writer problem**

**1) Reader/writer problem using semaphores**

**2) Reader-writer problem using monitors**

**3)Producer/consumer problem using threads/semaphores**

**4)producer consumer problem using  monitors**

5) 2-level pager table with LRU swapping

6) 2-level page table with FIFO swapping

**7) Car park problem using monitors**

**8) 2 producer and 2 consumer problem**

**9) Garden with two gates problem**

**10) Buddy page allocator**

**11) 3 threads in a deadlock state**

**12) Communication using named pipes**