SHRI VILEPARLE KELAVANI MANDAL'S
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC ACCREDITED with "A" GRADE (CGPA : 3.18)

A.Y.: 2023-24 **Class: S.Y.B.Tech** **Sub:** System Fundamentals

System Fundamentals Experiment List

Explore the internal commands of Linux and Write shell scripts to do the following:

1. **Display top 10 processes in descending order**

   ```
   ps aux --sort=-%cpu | head -n 11
   ```

2. **Display processes with highest memory usage.**

   ```
   ps aux --sort=-%mem | head -n 11
   ```

3. **Display current logged in user and logname.**

   ```
   echo "Current User: $(whoami)"
   echo "Login Name: $(logname)"
   ```

4. **Display current shell, home directory, operating system type, current path setting, current working directory.**

   ```
   echo "Current Shell: $SHELL"
   echo "Home Directory: $HOME"
   echo "Operating System Type: $(uname -s)"
   echo "Current PATH Setting: $PATH"
   echo "Current Working Directory: $(pwd)"
   ```

5. **Display OS version, release number, kernel version.**

   ```
   echo "OS Version: $(lsb_release -d | awk '{print $2, $3, $4}')"
   echo "Release Number: $(lsb_release -r | awk '{print $2}')"
   echo "Kernel Version: $(uname -r)"
   ```

6. **Write a command to display the first 15 columns from each line in the file**

   ```
   cut -c 1-15 filename
   ```

7. **cut specified columns from a file and display them**

   ```
   cut -f 2,4,7 filename
   ```

8. **Sort given file ignoring upper and lower case**

   ```
   sort -f filename
   ```

9. **Displays only directories in current working directory.**

   ```
   find . -maxdepth 1 -type d
   ```

10. Copying files from one place to another.

```
cp source_file destination
eg) cp example.txt backup/
```

11. moving files from one place to another.

```
mv source_file destination
eg) mv example.txt backup/
```

12. Removing specific directory with various options

```
rmdir directory_name (for empty directories)
rm -r directory_name (for directories and their contents)
```

13. list the numbers of users currently login in the system and then sort it.

```
who | awk '{print $1}' | sort
```

14. Merge two files into one file.

```
cat file1.txt file2.txt > merged_file.txt
```

15. changes the access mode of one file

```
chmod new_mode filename
eg) chmod 744 filename
```

16. display the last ten lines of the file.

```
tail -n 10 filename
```

17. to locate files in a directory and in a subdirectory.

```
find /path/to/directory -type f -name "filename_pattern"
eg) find /home/user/documents -type f -name "*.txt"
```

18. This displays the contents of all files having a name starting with ap followed by any number of characters.

```
cat ap*
cat /path/to/directory/ap* (specific directoy)
```

19. Rename any file aaa to aaa.aa1, where aa1 is the user login name.

```
mv aaa aaa.$(whoami)
```

Illustrate the use of sort, grep, awk, etc.

20. Write a command to search the word ‘picture’ in the file and if found, the lines containing it would be displayed on the screen.

```
grep 'picture' your_file.txt
```

21. Write a command to search for all occurrences of ‘Rebecca’ as well as ‘rebecca’ in file and display the lines which contain one of these words.

```
 grep -i 'Rebecca' your_file.txt
```

22. **Write a command to search all four-letter words whose first letter is a ‘b’ and last letter, a ‘k’.**

```
grep -w '^b..k$' your_file.txt
```

23. **Write a command to see only those lines which do not contain the search patterns**

```
grep -v 'pattern1\|pattern2' your_file.txt
```

24. **Implement various LRU cache/page replacement policy**

```
import Collection from OrderedDict
class LRUCache:
    def __init__(self, capacity):
        self.capacity = capacity
        self.cache = OrderedDict()

    def access_page(self, page):
        if page in self.cache:
            # Move the accessed page to the end
            self.cache.move_to_end(page)
        else:
            if len(self.cache) >= self.capacity:
                self.cache.popitem(last=False)

            # Add the new page to the cache
            self.cache[page] = None

    def display_cache(self):
        print("LRU Cache:", list(self.cache.keys()))

# Example usage
pages_lru = [1, 2, 3, 1, 4, 5, 2, 3, 6]
```

```
    lru_cache = LRUCache(3)


    for page in pages_lru:

        lru_cache.access_page(page)

        lru_cache.display_cache()
```

29. Implement various optimal cache/page replacement policy

```
class Optimalcache:
    def __init__(self, capacity):
        self.capacity = capacity
        self.cache = []

    def access_page(self, page):
        if page in self.cache:
            return

        if len(self.cache) < self.capacity:
            self.cache.append(page)
            return

        # Find the farthest page in the future that is not in the cache
        farthest_page = max(self.cache, key=lambda p: p not in
self.cache)

        # Replace the farthest page with the new page
        self.cache.remove(farthest_page)
        self.cache.append(page)

    def display_cache(self):
        print("Cache:", self.cache)

# Example usage
pages = [7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 1, 2, 0, 1, 7, 0, 1]
optimal_cache = OptimalCache(3)

for page in pages:
    optimal_cache.access_page(page)
    optimal_cache.display_cache()
```

30. Implement various FIFO cache/page replacement policy

```
class FIFOCache:
    def __init__(self, capacity):
        self.capacity = capacity
        self.cache = []

    def access_page(self, page):
        if page in self.cache:
            return

        if len(self.cache) < self.capacity:
            self.cache.append(page)
            return

        # Remove the first page from the cache
        self.cache.pop(0)

        # Add the new page to the cache
        self.cache.append(page)

    def display_cache(self):
        print("Cache:", self.cache)

# Example usage
pages = [7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 1, 2, 0, 1, 7, 0, 1]
fifo_cache = FIFOCache(3)

for page in pages:
    fifo_cache.access_page(page)
    fifo_cache.display_cache()
```

31. Implement FCFS CPU scheduling algorithm.

```
class FCFS:
    def __init__(self, processes):
        self.processes = processes

    def execute(self):
        for process in self.processes:
            print(f"Executing Process {process}")

processes_fcfs = [1,2,3,4,5]
fcfs_scheduler = FCFS(processes_fcfs)
fcfs_scheduler.execute()
```

32. Implement SJF CPU scheduling algorithm.

```
    class SOF:
        def init (self, processes):
            self.processes = processes

        def execute(self):
            self.processes.sort()
            for process in self.processes:
                print(f"Executing Process (process)")
    # Example usage
    processes_sif = = [5, 3, 1, 4, 2]
    sjf_scheduler = SJF(processes_sjf)
    sjf_scheduler.execute()
```

33. Implement Non Preemptive Priority CPU scheduling algorithm.

```
class PriorityNonPreemptive:
    def __init__(self, processes, priorities):
        self-processes = processes
        self.priorities = priorities
def execute(self):
    sorted_processes = [x for _, x in sorted(zip(self.priorities, self.processes))]
    for process in sorted_processes:

print(f"Executing Process {process}")
Example usage
processes _priority = [1, 2, 3, 4, 5]
priorities = [3, 1, 4, 2, 5]
priority_scheduler = PriorityNonPreemptive(processes_priority, priorities)
priority_scheduler.execute()
```

34. Implement Preemptive Priority CPU scheduling algorithm.

35. Implement SRTF CPU scheduling algorithm.

```
class SRTF:
    def __init__(self, processes, burst_times):
    self.processes = processes
    self.burst _times = burst _times

def execute(self):
    remaining_time = (process: burst for process, burst in zip(self.processes,
    self.burst_times))
    current_time = 0

    while remaining_time:
        available_processes = (p: t for p, t in remaining time.items() if t > 0 and current
    time >m t)
```

```
        if not available_ processes:
            current_time += 1
            continue

        shortest_process = min(available_processes, key-available_processes.get)
        print(f"Executing Process {shortest_process} at time {current_time}")
        remaining_time[shortest_process]-=1
        if remaining_time[shortest_process] == 0:
            del remaining_time[shortest_process]


        current_time += 1
Example usage
processes_srtf = ["P1", "P2", "P3", "P4", "P5"]
burst_times_srtf = [6, 8, 7, 3, 2]
srtf_scheduler = SRTF(processes_srtf, burst_times_srtf)
srtf_scheduler.execute()
```

36. Implement Round Robin CPU scheduling algorithm.

```
def round_robin(processes, burst_time, quantum):
    n = len(processes)
    remaining_time - burst_time.copy()
    waiting time = [0]*n
    turnaround time = [0] *n
    time = 0

    while True:
        done = True
        for i in range(n):
            if remaining_time[i] > 0:
            done = False
            if remaining_time[i] > quantum:
                time += quantum
            remaining_time[i] -= quantum
            else:
                time += remaining time[i]
                waiting time[i] = time - burst_time[i]
                remaining_time[i] =0
                turnaround_time[i] = time
        if done:
            break

    print("Process\twaiting Time\tTurnaround Time")
    for i in range(n):
    print(f"(processes[i]]\t(waiting_time[i]H\t\t(turnaround_time[i])")
# Example usage:
processes = [1, 2, 3|
burst_time=[10, 5, 8]
quantum = 2
round_robin(processes, burst_time, quantum)
```

37. Implement Best Fit Memory allocation policy.

```
def best_ fit(block_size, process_size):
    m = len(block_size)
    n = len(process_size)
    allocation = [-1] * n

    for i in range(n):
        best fit_idx = -1
        for j in range(m):
            if block_size[j] >= process_size[i]:
                if best_fit_idx == -1 or block_size[j] < block_size[best_fit_idx]:
                    best_fit_idx = j
        if best_fit_idx != -1:
            allocation[i] = best_fit_idx
            block_size[best_fit_idx] -= process_size[i]

    print("Process No.ItProcess Size\tBlock No.")
    for i in range(n):
        print(f"{i+1}\t\t(process_size[i])\t\t{allocation[i]+1 if allocation[i] != -1 else 'Not
    Allocated'}")
# Example usage:
block_size = [100, 500, 200, 300, 600]
process_size = [212, 417, 112, 426]
best_fit(block_size, process_size)
```

38. Implement First Fit Memory allocation policy.

```
def first_fit(memory_ blocks, process_sizes):
    allocation = [-1] * len(process_sizes)
    for i in range(len(process_sizes)):
        for j in range(len(memory_blocks)):
            if memory_blocks[j] >= process_sizes[i]:
            allocation[i] = j
            memory_blocks[j] -= process_sizes[i]
```

```
                break
    print("Process No.\tProcess Size\tBlock No.")
    for i in range(len(process_sizes)):
        print(f"(i + 1}\t\t {process_sizes[i]}\t\t ", end=""
        if allocation[i] l= -1:
            print(allocation[i] + 1)
        else:
            print("Not Allocated")
# Example usage:
memory_blocks = [100, 500, 200, 300, 600]
process_sizes = [212, 417, 112, 426]
first_fit(memory_blocks, process_sizes)
```

39. Implement Worst Fit Memory allocation policy.

```
def worst_fit(memory_blocks, process_sizes):
    allocation = [-1] * len(process_sizes)
    for i in range(len(process_sizes)):
        worst_index = -1
        for j in range(len(memory_blocks)):
            if memory_blocks[j] >= process_sizes[i]:
                if worst_index == -1 or memory_blocks[j] >
            memory_blocks[worst_index]:
                worst _index = j

            if worst index l= -1:
                allocation[i] = worst_index
                memory_blocks[worst_index] -= process_sizes[i]
    print("Process No. \tProcess Size\tBlock No.")
    for i in range(len(process_sizes)):
        print(f ”{i+1}\t\t{process_sizes[i]}\t\t”,end=”” )
        if allocation[i] != -1:
            print(allocation[i] + 1)
        else:
            print("Not Allocated")
Example usage:
memory_blocks = [100, 500, 200, 300, 600]
process_sizes = [212, 417, 112, 426]
worst_fit(memory_blocks, process_sizes)
```

40. Implement Producer -Consumer problem with Semaphore.

```
import threading
import time
from queue import Queue
buffer = Queue(maxsize=5)
```

```
mutex = threading.Semaphore(1)
full= threading.Semaphore(0)
empty = threading.Semaphore(buffer.maxsize)
def producer():
    for i in range(10):
        empty.acquire()
        mutex.acquire()
        item = f"Produced {i}"
        buffer.put(item)
        print(f"Produced: (item)")
        mutex.release()
        full.release()
        time.sleep(1)

def consumer():
    for i in range(10):
    full.acquire()
    mutex.acquire()
    item = buffer.get()
    print(f"Consumed: (item)")
    mutex.release()
    empty.release()
    time.sleep(1)

producer_thread = threading.Thread(target=producer)
consumer_thread = threading.Thread(target=consumer)

producer_thread.start()
consumer _thread.start()

producer_thread.join()
consumer_thread.join()
```

41. Implement order scheduling in supply chain using Banker's Algorithm

```
class BankerAlgorithm:
    def _init _(self, processes, resources):
    self.processes=processes
    self.resources=resources
    self.max_claim [[5, 5, 7), (3, 2, 2], [9, 0, 2], [2, 2, 2), [4, 3, 3]]
    self-allocation - [[0, 1, 0], [2, 0, 0], [3, 0, 2], [2, 1, 1], [0, 0, 2])
    self.need [[0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0])
    self.safe sequence = []
    self.work = resources.copy()
    self.finish - [False] * processes

    def calculate_need_matrix(self):
        for i in range(self.processes):
            for j in range(self.resources):
                self.need[i][j] = self.max_claim[i][j]-self.allocation[i][j]
    def is_safe_state(self):
        for i in range(self.processes):
            if not self.finish[i] and all(need <= self.work for need in self.need[i]]):
            self.work = [work + allocation for work, allocation in zip(self.work,
        self.allocation[i])]
            self.safe_sequence.append(i)
            self.finish[i] - True
            return self.is_safe_state()

        return all(self.finish)
    def run(self):
        self.calculate_need_matrix()
        if self.is_safe_state():
            print("Safe state found.")
            print("Safe Sequence:", self.safe_sequence)
        else:
            print("Unsafe state. No safe sequence found.")
Example usage:
banker BankerAlgorithm(processes=5, resources=3)
banker.run()
```

42. Implement FIFO Disk Scheduling Algorithms.

```
def fifo_disk_scheduling(requests, head):
    total_head_movements = 0
    current_head = head

    for request in requests:
        total_head_movements += abs(current_head - request)
        current_head = request
    return total_head_movements
# Example usage:
requests = [98, 183, 37, 122, 14, 124, 65, 67]
initial_head = 53
result = fifo_disk_scheduling(requests, initial_head)
print(f"FIFO Disk Scheduling Algorithm: Total head movements = {result}")
```

43. Implement SSTF Disk Scheduling Algorithms.

```
def sstf_disk_scheduling(requests):
    seek _count=0
    current _track = 0

    while requests:
        closest_request = min(requests, key=lambda x: abs(x - current_track))
        seek_count += abs(current_track - closest_request)
        current_track = closest_ request
        requests.remove(closest_request)

    return seek_count
# Example Usage
requests = [98, 183, 37, 122, 14, 124, 65, 67]
sstf_result = sstf_disk_scheduling(requests)
print(f"SSTF Disk Scheduling Seek Count: {sstf_result}")
```

44. Implement SCAN Disk Scheduling Algorithms.

```
def scan_disk_scheduling(requests, start_direction="left"):
    seek_count=0
    current_track = 0

    if start_ direction == "left":
        requests.sort()
    else:
        requests.sort(reverse=True)

    for request in requests:
        seek_count += abs(current_track - request)
        current_track = request
    return seek_count
#Example Usage
requests = [98, 183, 37, 122, 14, 124, 65, 67]
scan_result = scan_disk_scheduling(requests, start_direction="left"
print(f"SCAN Disk Scheduling Seek Count: {scan_result}")
```

45. Implement C-SCAN Disk Scheduling Algorithms.

```
def c_scan_disk_scheduling(requests, start_direction="left"):
    seek count=0
    current track = 0

    if start direction == "left":
        requests.sort()
    else:
        requests.sort(reverse=True)
    for request in requests:
        seek_count += abs(current_track - request)
        current_track = request

    seek_count += abs(current_track - (0 if start_direction == "left" else
    max(requests)))
    return seek_count
# Example usage
requests = [98, 183, 37, 122, 14, 124,65, 67]
c_scan_result = c_scan_disk_scheduling(requests, start_direction="left")
print(f"C-SCAN Disk Scheduling Seek Count: {c_scan_result}")
```

46. Implement Look Disk Scheduling Algorithms.

```
def look_disk_scheduling(requests, start_direction="left"):
    seek_count = 0
    current_track = 0

    if start_direction == "left":
        requests.sort()
    else:
        requests.sort(reverse=True)

    for request in requests:
        seek_count += abs(current_track - request)
        current_track = request
    return seek_count
# Example Usage
requests = [98, 183, 37, 122, 14, 124, 65, 67]
look_result = look_disk_scheduling(requests, start_direction="left")
print(f"Look Disk Scheduling Seek Count: {look_result}")
```

Implement Multithreading to create child processes using fork() system call.

47. Program where parent process sorts array elements in descending order and child process sorts array elements in ascending order.

```
#!/bin/bash

bubbleSort() {
  local array=("$@")
  local n=${#array[@]}

  for ((i = 0; i < n-1; i++)); do
    for ((j = 0; j < n-i-1; j++)); do
      if [[ ${array[j]} -lt ${array[j+1]} ]]; then
        # Swap
```

```
            temp=${array[j]}
            array[j]=${array[j+1]}
            array[j+1]=$temp
         fi
      done
   done

   echo "${array[@]}"
}

data=(12 5 7 18 3)
echo "Original Array: ${data[@]}"
{

   sorted_asc=($(bubbleSort "${data[@]}"))
   echo "Child process (Ascending Order): ${sorted_asc[@]}"
} &

sorted_desc=($(bubbleSort "${data[@]}"))
echo "Parent process (Descending Order): ${sorted_desc[@]}"
wait
```

48. **Program where parent process Counts number of vowels in the given sentence and child process will count number of words in the same sentence. The above programs should use UNIX calls like fork, exec and wait. And also show the orphan and zombie states.**

```
#!/bin/bash

echo “Enter a sentence:”
read sentence

count_vowels(){
echo “$1” | tr -cd ‘aeiouAEIOU’ | wc -c
}

if [ “$pid” -eq 0  ]; then
        exec echo “Child process (word count) : $(echo $sentence | wc -w)”
else
        wait $pid
        echo “Parent process (vowel count):$(count_vowels “$sentence”)”
fi
```

49. Write Shell script to copy files from one folder to another

```
#!/bin/bash

echo "Enter source folder:"
read source_folder

echo "Enter destination folder:"
read destination_folder

cp -r "$source_folder"/* "$destination_folder"
echo "Files copied successfully."
```

50. Write Shell script Count number of words, characters and lines.

```
#!/bin/bash
echo "Enter file name"
read f
echo number of lines
wc -l <$f
echo No. of words
wc -w <$f
echo No. of characters
wc -c <$f
```

52. Write Shell script To describe files in different format.

```
#!/bin/bash
echo Enter directory name
read d
echo menu
echo 1.Short hand descriptor
echo 2.Long hand descriptor
echo enter your choice
read a
```

```
case $a in
1) ls $d
;;
2) ls -l $d
;;
3) ls -a $d
;;
*)echo invalid choice
;;
esac
```

53. **Write Shell script to find factorial of given number using bash script**

```
#!/bin/bash
echo enter a number
read a
let fact=1
while [ $a -gt 0 ]
do
let fact=fact\*a
let a=-1
done
echo Factorial is $fact
```

54. **Display first 10 natural numbers using bash script**

```
#!/bin/bash

for ((i=1; i<=10; i++));
do
echo "$i"
done
```

55. **Display Fibonacci series using bash script**

```
#!/bin/bash

echo "Enter the number of terms: "
read n

a=0
b=1

echo "Fibonacci series:"
for ((i = 0; i<n; i++)); do
  echo -n "$a "
  next=$((a + b))
  a=$b
  b=$next
done
echo
```

56. **Find given number is prime or nor using bash script**

```
#!/bin/bash


echo "enter a number: "

read num


if [ "$num" -lt 2 ]; then

  echo "$num is not prime."

  exit

fi


is_prime=1


for ((i=2; i * 1 <= num; i++)); do

  if [ $((num % i)) -eq 0 ]; then

    is_prime=0

    break;

  fi

done


if [ "$is_prime" -eq 1 ]; then

  echo "$num is prime."

else

  echo "$num is not prime."

fi
```

57. **Write shell script to finding biggest of three numbers**

```
#!/bin/bash

echo "enter three numbers: "

read num1 num2 num3
```

```
max=$num1
if [ "$num2" -gt "$max" ]; then
  max=$num2
fi
if [ "$num3" -gt "$max" ]; then
  max=$num3
fi
echo "the biggest number is: $max"
```

58. **Write shell script to reversing a number**

```
#!/bin/bash
echo "enter a number: "
read num
reversed=0
while [ "$num" -ne 0 ]; do
        remainder=$((num % 10))
        reversed=$((reversed * 10 + remainder))
        num=$((num / 10))
done
echo "reversed number: $reversed"
```

59. **Write shell script find Sum of individual digits (1234 => 1+2+3+4=10)**

```
#!/bin/bash
echo "enter a number: "
read num
while [ "$num" -ne 0]; do
  remainder=$((num % 10))
  sum=$((sum + remainder))
  num=$((num / 10))
done
echo "sum of individual digits: $sum"
```