Operating system

Assignment-1

========================================================

Task 1: Process Creation Utility

========================================================

Python Code:

------------

def create\_child\_processes(n):

processes = []

for i in range(n):

pid = os.fork()

if pid == 0: # Child process

print(f"[Child {i+1}] PID={os.getpid()}, PPID={os.getppid()}, Message=Hello from child {i+1}")

os.\_exit(0)

else:

processes.append(pid)

for pid in processes:

os.waitpid(pid, 0)

Sample Output:

--------------

Enter number of child processes: 3

[Child 1] PID=2011, PPID=2010, Message=Hello from child 1

[Child 2] PID=2012, PPID=2010, Message=Hello from child 2

[Child 3] PID=2013, PPID=2010, Message=Hello from child 3

Complexity:

-----------

Time Complexity: O(n)

Space Complexity: O(n)

========================================================

Task 2: Command Execution Using exec()

========================================================

Python Code:

------------

def execute\_command\_in\_children(commands):

processes = []

for i, cmd in enumerate(commands):

pid = os.fork()

if pid == 0:

print(f"[Child {i+1}] PID={os.getpid()} executing: {cmd}")

os.execvp(cmd[0], cmd) # Replace child with command

else:

processes.append(pid)

for pid in processes:

os.waitpid(pid, 0)

Sample Output:

--------------

[Child 1] PID=2050 executing: ['ls', '-l']

-rwxr-xr-x 1 user user 3456 Sep 15 process\_management.py

[Child 2] PID=2051 executing: ['date']

Mon Sep 15 15:42:11 IST 2025

[Child 3] PID=2052 executing: ['ps', '-f']

UID PID PPID C STIME TTY TIME CMD

user 2052 2050 0 15:42 pts/0 00:00:00 ps -f

Complexity:

-----------

Time Complexity: O(n)

Space Complexity: O(1) (exec replaces child)

========================================================

Task 3: Zombie & Orphan Processes

========================================================

Python Code:

------------

def zombie\_process():

pid = os.fork()

if pid == 0:

print(f"[Child] PID={os.getpid()}, PPID={os.getppid()} exiting immediately")

os.\_exit(0)

else:

print(f"[Parent] PID={os.getpid()} created child PID={pid}")

print("Parent not calling wait() → child becomes ZOMBIE")

time.sleep(10) # Check with 'ps -el | grep defunct'

def orphan\_process():

pid = os.fork()

if pid == 0:

time.sleep(5)

print(f"[Orphan Child] PID={os.getpid()}, new PPID={os.getppid()}")

os.\_exit(0)

else:

print(f"[Parent] PID={os.getpid()} exiting early → ORPHAN created")

os.\_exit(0)

Sample Output:

--------------

Zombie Example:

[Parent] PID=2100 created child PID=2101

Parent not calling wait() → child becomes ZOMBIE

(ps -el | grep defunct shows child in defunct state)

Orphan Example:

[Parent] PID=2120 exiting early → ORPHAN created

[Orphan Child] PID=2121, new PPID=1

Complexity:

-----------

Time Complexity: O(1)

Space Complexity: O(1)

========================================================

Task 4: Inspecting Process Info from /proc

========================================================

Python Code:

------------

def inspect\_process(pid):

status\_file = f"/proc/{pid}/status"

exe\_file = f"/proc/{pid}/exe"

fd\_dir = f"/proc/{pid}/fd"

try:

with open(status\_file, "r") as f:

for line in f:

if line.startswith("Name:") or line.startswith("State:") or line.startswith("VmSize:"):

print(line.strip())

exe\_path = os.readlink(exe\_file)

print(f"Executable Path: {exe\_path}")

print("Open File Descriptors:")

for fd in os.listdir(fd\_dir):

try:

target = os.readlink(os.path.join(fd\_dir, fd))

print(f" FD {fd}: {target}")

except:

print(f" FD {fd}: [No access]")

except FileNotFoundError:

print(f"PID {pid} does not exist.")

except PermissionError:

print(f"Permission denied for PID {pid}.")

Sample Output:

--------------

Enter PID to inspect: 2135

Name: bash

State: S (sleeping)

VmSize: 16284 kB

Executable Path: /usr/bin/bash

Open File Descriptors:

FD 0: /dev/pts/0

FD 1: /dev/pts/0

FD 2: /dev/pts/0

Complexity:

-----------

Time Complexity: O(m) (m = number of lines in status + FDs)

Space Complexity: O(1)

========================================================

Task 5: Process Prioritization with nice()

========================================================

Python Code:

------------

def cpu\_intensive\_task(child\_id, duration=5):

start = time.time()

count = 0

while time.time() - start < duration:

count += 1

print(f"[Child {child\_id}] PID={os.getpid()}, Finished count={count}")

def priority\_demo():

processes = []

nice\_values = [-5, 0, 5, 10]

for i, nice\_val in enumerate(nice\_values, start=1):

pid = os.fork()

if pid == 0:

try:

os.nice(nice\_val)

except PermissionError:

print(f"[Child {i}] Cannot decrease nice without sudo, using default")

print(f"[Child {i}] PID={os.getpid()}, Nice={os.nice(0)} started")

cpu\_intensive\_task(i)

os.\_exit(0)

else:

processes.append(pid)

for pid in processes:

os.waitpid(pid, 0)

Sample Output:

--------------

[Child 1] PID=2200, Nice=-5 started

[Child 2] PID=2201, Nice=0 started

[Child 3] PID=2202, Nice=5 started

[Child 4] PID=2203, Nice=10 started

[Child 1] PID=2200, Finished count=104323392

[Child 2] PID=2201, Finished count=82539201

[Child 3] PID=2202, Finished count=65320412

[Child 4] PID=2203, Finished count=49839010

Observation:

------------

- Lower nice (-5) = higher priority = more CPU cycles

- Higher nice (10) = lower priority = fewer CPU cycles

Complexity:

-----------

Time Complexity: O(n \* t), where t = loop duration

Space Complexity: O(1) per child