Name: Monika Raghav

Roll.No.:2301420010

Course: B.Tech(CSE) Data Science

ASSIGNMENT-4

1.Batch Processing Simulation:

Write a python script to execute multiple.py files sequentially, mimicking batch processing.

Code:

```
  GNU nano 8.4
import subprocess

# List of Python scripts to execute sequentially
scripts = ['script1.py', 'script2.py', 'script3.py']

for script in scripts:
    print(f"\n=== Executing {script} ===")
    result = subprocess.run(['python3', script])
    if result.returncode == 0:
        print(f"{script} executed successfully.\n")
    else:
        print(f"Error while executing {script}.\n")
```

Output:

```
=== Executing script1.py ===
Running Script 1
script1.py executed successfully.


=== Executing script2.py ===
Running Script 2
script2.py executed successfully.


=== Executing script3.py ===
Running Script 3
script3.py executed successfully.
```

2.System Startup And Logging

Code:

```python
startup_simulation.py
Simulate system startup: create multiple processes (multiprocessing) which run simple tasks,
and log lifecycle events (start, end, elapsed) into process_log.txt
"""
import logging
import multiprocessing as mp
import time
import os

LOGFILE = "process_log.txt"

def setup_logging():
    logging.basicConfig(
        filename=LOGFILE,
        level=logging.INFO,
        format="%(asctime)s [%(processName)s:%(process)d] %(levelname)s: %(message)s"
    )
    logging.getLogger().addHandler(logging.StreamHandler())  # also print to console

def worker(name, duration):
    logging.info(f"Worker {name} starting (PID={os.getpid()})")
    t0 = time.time()
    # simulate work
    time.sleep(duration)
    elapsed = time.time() - t0
    logging.info(f"Worker {name} finished (PID={os.getpid()}) elapsed={elapsed:.2f}s")

def main():
    setup_logging()
    logging.info("Startup simulation beginning")

    # Configure simulated startup tasks: (name, duration seconds)
    tasks = [
        ("init_services", 2),
        ("mount_filesystems", 1),
        ("network_manager", 3),
        ("login_manager", 1.5),
        ("cron_jobs", 0.8),
    ]
```

Output:

```
# view log
cat process_log.txt

Startup simulation beginning
Worker init_services starting (PID=28564)
Worker mount_filesystems starting (PID=28565)
Worker network_manager starting (PID=28566)
Worker login_manager starting (PID=28567)
Worker cron_jobs starting (PID=28576)
Worker mount_filesystems finished (PID=28565) elapsed=1.00s
Worker cron_jobs finished (PID=28576) elapsed=0.80s
Worker init_services finished (PID=28564) elapsed=2.00s
Worker login_manager finished (PID=28567) elapsed=1.50s
Worker network_manager finished (PID=28566) elapsed=3.00s
Startup simulation complete
2025-11-10 15:29:39,555 [MainProcess:28563] INFO: Startup simulation beginning
2025-11-10 15:29:39,560 [init_services:28564] INFO: Worker init_services starting (PID=28564)
2025-11-10 15:29:39,765 [mount_filesystems:28565] INFO: Worker mount_filesystems starting (PID=28565)
2025-11-10 15:29:39,968 [network_manager:28566] INFO: Worker network_manager starting (PID=28566)
2025-11-10 15:29:40,171 [login_manager:28567] INFO: Worker login_manager starting (PID=28567)
2025-11-10 15:29:40,377 [cron_jobs:28576] INFO: Worker cron_jobs starting (PID=28576)
2025-11-10 15:29:40,767 [mount_filesystems:28565] INFO: Worker mount_filesystems finished (PID=28565) elapsed=1.00s
2025-11-10 15:29:41,179 [cron_jobs:28576] INFO: Worker cron_jobs finished (PID=28576) elapsed=0.80s
2025-11-10 15:29:41,561 [init_services:28564] INFO: Worker init_services finished (PID=28564) elapsed=2.00s
2025-11-10 15:29:41,673 [login_manager:28567] INFO: Worker login_manager finished (PID=28567) elapsed=1.50s
2025-11-10 15:29:42,969 [network_manager:28566] INFO: Worker network_manager finished (PID=28566) elapsed=3.00s
2025-11-10 15:29:42,971 [MainProcess:28563] INFO: Startup simulation complete
```

3. System Calls and IPC

Code:

```python
#!/usr/bin/env python3
"""
ipc_pipe_fork.py
Simple IPC using os.pipe() and os.fork():
Parent sends a message to child using the pipe, child reads it and responds.
"""
import os
import sys

def parent_child_communication():
    # create a pipe: r, w are file descriptors
    r, w = os.pipe()

    pid = os.fork()
    if pid == 0:
        # child process
        os.close(w)  # close write end in child
        rfd = os.fdopen(r, 'r')
        msg = rfd.read()  # read everything the parent writes
        print(f"Child (pid {os.getpid()}): received from parent: {msg.strip()}")
        rfd.close()
        sys.exit(0)
    else:
        # parent process
        os.close(r)  # close read end in parent
        wfd = os.fdopen(w, 'w')
        message = "Hello child! This is parent.\n"
        wfd.write(message)
        wfd.flush()
        wfd.close()
        # wait for child to finish
        pid_done, status = os.waitpid(pid, 0)
        print(f"Parent: child {pid_done} exited with status {status}")

if __name__ == "__main__":
    parent_child_communication()
```

Output:

```
Child (pid 31786): received from parent: Hello child! This is parent.
Parent: child 31786 exited with status 0
```

4. VM Detection and Shall Interaction

Code:

```bash
GNU nano 8.4                                                                          system_info.sh *
#!/usr/bin/env bash
# system_info.sh - print system details (some commands may need sudo)

echo "=== uname -a ==="
uname -a
echo

echo "=== lscpu ==="
lscpu
echo

echo "=== free -h ==="
free -h
echo

echo "=== lsblk ==="
lsblk
echo

echo "=== ip addr (show interfaces) ==="
ip -c addr
echo

echo "=== last reboot (uptime) ==="
uptime
echo

# dmidecode may require root privileges; print a short note if not accessible
if command -v dmidecode >/dev/null 2>&1; then
    echo "=== dmidecode -t system (requires sudo) ==="
    if [ "$(id -u)" -eq 0 ]; then
        dmidecode -t system
    else
        echo "dmidecode available but requires sudo. Run: sudo dmidecode -t system"
    fi
else
    echo "dmidecode not installed or not available."
fi

echo
echo "=== lspci (if available) ==="
if command -v lspci >/dev/null 2>&1; then
    lspci | head -n 20
else
    echo "lspci not available"
```

Output:

```
=== uname -a ===
Linux kali 6.12.33+kali-amd64 #1 SMP PREEMPT_DYNAMIC Kali 6.12.33-1kali1 (2025-06-25) x86_64 GNU/Linux

=== lscpu ===
Architecture:             x86_64
  CPU op-mode(s):         32-bit, 64-bit
  Address sizes:          39 bits physical, 48 bits virtual
  Byte Order:             Little Endian
CPU(s):                   1
  On-line CPU(s) list:    0
Vendor ID:                GenuineIntel
  Model name:             11th Gen Intel(R) Core(TM) i7-1165G7 @ 2.80GHz
    CPU family:           6
    Model:                140
    Thread(s) per core:   1
    Core(s) per socket:   1
    Socket(s):            1
    Stepping:             1
    BogoMIPS:             5606.39
    Flags:                fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush mmx fxsr sse sse2 ht syscall nx rdtscp lm constant_tsc rep_good nopl xtopolog
                          ssse3 fma cx16 pcid sse4_1 sse4_2 movbe popcnt aes xsave avx f16c rdrand hypervisor lahf_lm abm 3dnowprefetch ibrs_enhanced fsgsbase bmi1 avx2 bmi2 invpcid r
                          1d arch_capabilities
Virtualization features:
  Hypervisor vendor:      KVM
  Virtualization type:    full
Caches (sum of all):
  L1d:                    48 KiB (1 instance)
  L1i:                    32 KiB (1 instance)
  L2:                     1.3 MiB (1 instance)
  L3:                     12 MiB (1 instance)
NUMA:
  NUMA node(s):           1
  NUMA node0 CPU(s):      0
Vulnerabilities:
  Gather data sampling:   Not affected
  Indirect target selection: Mitigation; Aligned branch/return thunks
  Itlb multihit:          Not affected
  L1tf:                   Not affected
  Mds:                    Not affected
  Meltdown:               Not affected
  Mmio stale data:        Not affected
  Reg file data sampling: Not affected
  Retbleed:               Mitigation; Enhanced IBRS
  Spec rstack overflow:   Not affected
  Spec store bypass:      Vulnerable
  Spectre v1:             Mitigation; usercopy/swapgs barriers and __user pointer sanitization
  Spectre v2:             Mitigation; Enhanced / Automatic IBRS; PBRSB-eIBRS SW sequence; BHI SW loop, KVM SW loop
  Srbds:                  Not affected
```