2301010218

Amit

Operating System Assignment - 5

Q:- Necessity of operating system and essential abstraction.

Ans Modern computer system still heavily on OS despite advantages in hardware because OS serves two crucial, non-redundant roles:

- Resources Management - It manages and detects & allocates system resources (CPU memory, I/O device) efficiency.

- Abstraction layers: It provides a symple clean and consistent interfaces to the underlying complex, raw hardware, making it easier for programmers to write applications.

Q:- Comparison of OS Structure:

Ans:

Feature	Monolithic	Layered	Microkernal
Structure	All OS services are combined into binary running in kernal space.	Services are organized in layers, with each layer relying only one the funcns of the layers	The kernal is minimal only handling core service. Other service run as user level server.
Speed.	Fasters execution due to minimal over head	Slower than monolithic due to the overhead of inter layer communication	Slowest because most service calls require context and inter-Process.
Reliability	Low	Moderate	Highest

3. Analysis of thread Vs Process Efficiency:

Feature	Process	Threads	Efficiency
Creation	Slow. Requires allocating new virtual address space a new process control Block, and copying data structure	Fast. Only requires creating a new thread control Block	Threads are more efficient.
Context Switching	Slow. The OS must save the entire state of the old process and load the new one	Fast. The OS only needs to save registers and the stack pointer.	Thread are more efficient.
Resource Usage	High. Each process requires its own private, isolated copy of resources.	Low. Threads share code, data, files and heap memory of the parent process.	Threads are more efficient in terms of system memory footprint.

4. Memory allocation simulation (First-fit, Best-fit, Worst-fit):

Ans

Process	Requirement
P_1	12 MB
P_2	18 MB
P_3	6 MB

Algorithm

First-Fit

Allocation Steps

P_1 (12MB): Allocates the first Block that fits (20MB), remaining 8 MB P_2 (18MB): No Block larg enough. wait (P_3 6MB): Fits (10MB). Remaing: 4MB

Total unused space 8MB + 4MB + 5MB = 17MB

P_2 is waiting

Best-fit

P_1 (12MB): Allocates the smallest block that fits

P_2 (18MB): No block large enough

P_3 (6MB): Allocates the smallest block that fits

Total unused space: $8 + 4 + 5 = 17$ mb

P_2 is waiting

Worst-fit

P_1 (12MB): Allocates the largest block

P_2 (18MB): No blocks large enough

P_3 (6MB): Allocates the largest remaing block

Total unused space: $8 + 4 + 5 = 17$ MB

P_2 is waiting

Q (1) First come first served (FCFS)

-> Process are executed in the order of their cervical time.

- P_1 arrives at 0, executes for 5ms (0-5)
- P_2 " " 1, but waits until P_1 finishes, execute for 3ms (5-8)
- P_3 " " 2, wait untill P_2 finishes, execute for 8ms (8-16)
- P_4 " " 3, waits untill P_3 finishes, " " 3ms (16-19)

P_1	P_2	P_3	P_4	
0	5	8	16	19

(2) Shortest JOB first (SJF):

- Processes are selected based on the shortest Burst Time among those that have arrived:
- $t = 0$, only P_1 has arrived. P_1 starts (BT=5)
- $t = 5$, P_1 finishes P_2 (At=1, Bt=3), P_3 (At=2, Bt=8) and P_4 (at=3, Bt=3) have all arrived.
- Selection: P_2 and P_4 both have BT=3. Choose P_2 (arrived first). P_2 executes.
- $t = 8$ P_2 finishes: P_3 (BT=8) remains, P_3 executes (11-19)

P_1	P_2	P_4	P_3	
0	5	8	11	19

3. Round - Robin - Quantum = 4 ms
Processes are given a maximum of 4 ms CPU time, if not finished they are preempted and moved to the back of the ready queue.

Time (ms)	Executing Process	Remaing BT	Ready Queue
0-4	P_1 (BT = 5)	1	P_2(at 1), P_3(at 2), P_4
4-7	P_2 (BT = 3)	0 (Finished)	P_3, P_4, P_1 (remaining BT
7-11	P_3 (BT = 8)	4	P_4, P_1, P_3 (rema BT=)
11-14	P_4 (BT = 3)	0 (Finished)	P_1, P_3
14-15	P_1 (BT = 1)	0 (Finished)	P_3
15-19	P_3 (BT = 4)	0 (Finished)	ϕ

P_1	P_2	P_3	P_4	P_1	P_3	
0	4	7	11	14	15	19

B. Calculating of Avg. Time

1. FCFS

Processes	BT	AT	Completion Time	TAT	WT
P_1	5	0	5	5	0
P_2	3	1	8	7	4
P_3	8	2	16	14	6
P_4	3	3	19	18	13
Sum				42	23

* Avg TAT = 42/4 = 10.5 ms
* Avg WT = 23/4 = 5.75 ms

2. SJF

Process	BT	AT	Completion Time	TAT	WT
P_1	5	0	5	5	0
P_2	3	1	8	7	4
P_3	8	2	19	17	6
P_4	3	3	11	19	13
Sum				37	23

* Avg TAT = 42/4 = 10.5 ms
* Avg WT = 23/4 = 5.75 ms

3 Round- Robin

Process	BT	AT	Completion Time	TAT	WT
P_1	5	0	15	15	10
P_2	3	1	7	6	3
P_3	8	2	19	17	9
P_4	3	3	14	11	8
Sum				49	30

* Avg TAT = 12.25 ms
Avg WT = 7.5 ms

Q Distributed file Management

Ans a. Issues in distributed OS design for file sharing and resource management.

1. Global clock Inconsistency: Distributed systems lack a single, synchronized clock. This makes it challenging to establish a consistent order of events, which is critical for file consistency and clock ordering.

2. Network latency and Partitioning: The network introduced delay after every file access, severly degrading performance. A network partition can split the system, leading to conflicting update.

3. Authentication and authorization: Securing resource is complex decuse control risk must be consistency and organized across multiple, independent server.

4. Resource contention and deadlock: Distributed processes can contain non-shared network resource.

Q. Synchonous Checkpointing and recovery

Ans 4 Initialization: A checkpoint initiator (Process P_i) begins the process by recording its local state and sending a checkpoint request message to all other process.

2. Coordination: Upon receiving the request, all other process (P_i) immediately block their normal execution and stop sending application messages.

3. Local State Recording: Each blocked process records its current local state and the state of its communication channels.

4. Commit: Once all process confirm they have recorded their state they realise their blocks and resume execution. The collection of all local states from the single consistent global checkpoint.

Q. IoT Smart Home System

Ans a. Process scheduling Strategy

- Algorithm Selection: Preemptive Priority Scheduling with interrupt-driven activation.
- Justification and Strategy:

1. Priority Assignment: Assign the highest priority to the security Device Interupts. All other task recieve lower Priority.

2. Preemption: Use preemption if a high-priority security Interupt occurs while a low-priority task is running, the scheduler will immediately suspend the lower-priority task and dispatch the critical security Interupt handler.

3. Justification: This ensures minimal latency for life-safety and security-critical tasks, fulfilling the primary requirement of prioritizing Interupts. The low priority-task can tolerate a slight delay, but security alerts cannot.

Q 12. Python System Call Implementation:

Ans -> Selected OS and System call

- Operating System = Linux
- Basic System Call demonstration: File mangement

Code:

```
import os
import stat
import time
FILE_NAME = "syscall_demo_file.txt"
FILE_CONTENT = "This is a text of the "write" system call in python".
def demonstrate_system_calls.():
try :
Print(f"→ calling 'open' system call to create and open: {file_open}
    file_descriptor = os.open (
        Path = file_Name,
        flags = OS.O_CREATE | OS.O_WRONGILY | OS.O_TRUNC,
        mode = 0.644.
    )
    Print(f" Success, file descriptor (FD): {file_descriptor}")
    Print(f"→ calling 'close' system call on FD: {" } ")
    Print ("Success. file closed")
Print(f"/n verification: The file {file_Name}" has been
check and except OSError as e:
    Print (f"\nAn OS error occurred : {e}")
finally :
    Print (f"/n → calling 'unlike' system call to delete: {
    file_Nam}
    Print ("Success. file deleted")
if __name__ = "__main__" :
    demonstrate_system_calls()
```