[Question 1]

*1 Identify two advantages and two disadvantages of open-source operating systems. Identify the*

*types of people who would find each aspect to be an advantage or a disadvantage.*

Advantages:

**Transparency**  
- Anyone can access the source code of open-source software.   
- It is possible to examine the code for flaws, malicious elements, and quality control.   
  
  
**Stability**  
- Increased dependability and stability over time.

- Bugs are found through testing and development that is driven by the community.

Disadvantages:

**Compatibility**  
- Restricted support for specific hardware setups, which makes it difficult to locate drivers or get the best performance on particular systems.   
  
**Insufficient Support from Vendors**   
- May not have committed vendor support, which could cause problems getting expert help

* lengthens the time it takes to resolve important problems.

*2 Describe two challenges of designing operating systems for mobile devices compare with designing operating system for traditional PCs.*

**Segmentation by device manufacturers**

- Segmentation or customization by device manufacturers

like Samsung, Google (Pixel devices), LG, HTC, etc.

- The Linux kernel, which forms the basis of the Android operating system, can be modified by manufacturers to enhance performance, incorporate proprietary drivers, or add support for particular hardware features.

- The makers and models of devices can differ in these kernel alterations

- This may result in fragmentation, which would provide difficulties for users, developers, and the ecosystem as a whole if multiple operating system versions and variations appear on different devices.

**Form Factors**

- There are many distinct types of mobile devices, such as tablets, smartphones, and wearables, and each has a unique screen size, resolution, and input method.

- It's difficult to design an operating system that can accommodate these many form factors while preserving a consistent user experience across platforms.

*3) Rank the following storage systems from fastest to slowest*

|  |  |  |
| --- | --- | --- |
| RANK | Storage system | explanation |
| 1 | Registers | In CPU, data holding place to execute programs and operations |
| 2 | Cache | Buffer between CPU and main memory. |
| 3 | Main Memory (RAM) | volatile memory, store CPU instructions |
| 4 | Nonvolatile Memory | retains data even when the power is turned off |
| 5 | Optical Disk | optical technology, archival purposes |
| 6 | Hard-Disk Drives (HDDs) | mechanical latency and seek times. |
| 7 | Magnetic Tapes | archival purposes and long-term data backup |

[Question 2]

1) What are the two models of interprocess communication? (2)

Message Passing:

- used by most operating systems.

- Multiple processes can read and write data to the message queue without being connected to each other

- Each process has its own address space, and communication occurs through message queues, shared memory, or sockets

Shared Memory:

- is a feature of Windows and POSIX operating systems

- this allows many processes to access shared memory at the same time.

- multi-threaded applications and parallel computing environments frequently employ this.

2) What are the strengths and weaknesses of the two models of interprocess communication requested in 1) above? (3)

Message Passing:

|  |  |
| --- | --- |
| strengths | weaknesses |
| Isolation  - offers robust isolation between processes or threads (each process or thread has its own private memory space)  - enhances system security and fault tolerance by preventing unintended access to shared resources | Complexity  - Message queues, message formats, error-handling techniques, and network protocols must all be handled carefully when implementing message-passing communication. |
| Portable + scalable  - does not depend on underlying hardware or operating system | Overhead  - Message passing communication often incurs overhead in terms of message creation, serialization, deserialization, and transmission. – this overhead can reduce performance, increase latency, and require more system resources. |
| Asynchronous Communication  - event-driven and non-blocking programming paradigms are made possible, which permits asynchronous communication between processes. | Restricted access  - prevents direct access to the data or memory layout, it might be more constrained and restrictive.  - can also place restrictions on the size, frequency, and order of messages. |

Shared Memory:

|  |  |
| --- | --- |
| strengths | weaknesses |
| Efficiency  - allows processes or threads to directly access the same physical memory regions, it provides fast data transfer between them. | all processes must ensure that they are not writing to the same memory address. |
| Low latency  - eliminates the need for message queuing, buffering, and network communication overhead, it often has a lower latency than message passing. | Addressing issues like synchronization and memory protection is necessary when using a shared memory approach. |
| Resource efficiency  - Good for applications that require shared access to a large pool of memory. | Vulnerabilities in shared memory systems could be used by malicious programs or attackers to undermine system confidentiality or integrity. |

3) Explain why Java programs running on Android systems do not use the standard Java API and virtual machine.

Firstly, there are differences in the underlying architecture and requirements of the Android platform (Android uses a different virtual machine called Dalvik or the more recent Android Runtime (ART) whereas traditional environments use Java Virtual Machine (JVM)).

Although Android apps are written in Java, the Dalvik virtual machine—rather than the JVM—is used to run them. Android applications can operate on several underlying architectures without requiring recompilation thanks to the Dalvik virtual machine.

While Android applications have certain requirements and limits related to mobile devices, such as restricted memory, considerations for battery life, and a variety of hardware combinations, the standard Java API and JVM are optimized for desktop and server contexts.

[Question 3]

1) Including the initial parent process, how many processes are created by the following program?

|  |  |
| --- | --- |
| iteration |  |
| 1 | P fork()  ├── P  └──C1  2 processes (1 parent, 1 child) |
| 2 | P fork()  ├── P fork()  ├── P  └──C2  └──C1  4 processes (1 parent, 3 children) |
| 3 | P fork()  ├── P fork()  ├── P fork()  ├── P  └──C3  └──C2  └──C1  8 processes (1 parent, 7 children) |
| 4 | P fork()  ├── P fork()  ├── P fork()  ├── P fork()  ├── P  └──C4    └──C3  └──C2  └──C1  16 processes (1 parent, 15 children) |
| 5 | P fork()  ├── P fork()  ├── P fork()  ├── P fork()  ├── P fork()  ├── P  └──C5  └──C4    └──C3  └──C2  └──C1  32 processes (1 parent, 31 children) |

2) Using the following program, explain what the output will be at lines X and Y.

pid = fork();

\\ invoking fork() branches to create a parent and a child process

\\ when invoked, the value of fork() is stored in the pid variable

\\ the program has two execution paths

\\\\\\\\\\\\\\\\\\\\\ PATH 1

\\ If pid is 0, the program is executing in the child process.

if (pid == 0) {

for (i = 0; i < SIZE; i++) {

nums[i] \*= -1;

printf("CHILD: %d ", nums[i]); /\* LINE X \*/

}

}

\\ OUTPUT: CHILD: 0 CHILD: -2 CHILD: -4 CHILD: -6 CHILD: -8

\\\\\\\\\\\\\\\\\\\\\ PATH 2

\\ If pid is greater than 0, the program is executing in the parent process

} else if (pid > 0) {

wait(NULL);

for (i = 0; i < SIZE; i++)

printf("PARENT: %d ", nums[i]); /\* LINE Y \*/

}

\\ OUTPUT: PARENT: 0 PARENT: -2 PARENT: -4 PARENT: -6 PARENT: -8

At line /\* LINE X \*/, the output wil be:

CHILD: 0 CHILD: -2 CHILD: -4 CHILD: -6 CHILD: -8

At line /\* LINE Y \*/the output wil be:

PARENT: 0 PARENT: -2 PARENT: -4 PARENT: -6 PARENT: -8

3) Describe the actions taken by a kernel to context-switch between processes.

An operating system that supports multitasking uses a technique called a context switch to change the execution context of one process to another.

- the state of the current process is stored (the “old” process)

- the next process to run is selected

- the state of the next process to be executed is loaded (the “new” process)

- the next process is executed starting the instruction cycle from the new process's program counter.

[Question 4]

1) Using Amdahl’s law, calculate the speedup gain for the following applications:

Amdahl's law predicts the maximum potential speedup of a program when parallelizing a portion of it:

Where:

is the fraction of the program that can be parallelized.

is the number of processing cores

a. 50 percent parallel with

(a) eighteen processing cores

and (b) six processing cores

b. 67 percent parallel with

(a) seven processing cores

and (b) two processing cores

c. 80 percent parallel with

(a) four processing cores

and (b) eight processing cores

2) Consider the following code segment:

(a) How many unique processes are created?

5 processes (1 parent, 4 children)

(b) How many unique threads are created?

1 Thread are created.

|  |  |
| --- | --- |
| iteration |  |
| 1 | P fork()  └──C1 fork(), thread\_create()  ├── T3  ├── C4  └──C2  └──C5  5 processes (1 parent, 4 children)  1 Thread |

Therefore 5 processes (1 parent, 4 children) & 1 Thread are created.

3) Determine if the following problems exhibit task or data parallelism:

a. Using a separate thread to generate a thumbnail for each photo in a collection

- when a thumbnail is generated, it is a task

- each task can be performed concurrently.

- each task can be assigned a thread, data does not need to be shared between threads

- this is task parallelism

b. Transposing a matrix in parallel

- each matrix element can be assigned a thread

- each matrix element will be have the same operation performed

- each matrix element will be separated into subsets, with each subset is processed concurrently by different threads

- this is data parallelism

c. An ETF application that uses different threads to extract data from multiple data sources, transform the extracted data and load the transformed data in the same database

- an ETF application has specific tasks, mainly extracting data, transforming data & loading data

- each task can be performed concurrently.

- each task can be assigned a thread, although operation can be performed on different data subsets where data is shared between threads.

- this is task & data parallelism

The following processes are being scheduled using a preemptive, priority-based, round-robin scheduling algorithm:

a. Show the scheduling order of the processes using a Gantt chart.

Gantt chart

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Time | 0 | 10 | 20 | 30 | 40 | 50 | 60 | 70 | 80 | 90 | 100 | 110 | 115 | 120 |
| Process | P2 | P2 | P3 | P3 | P4 | P4 | P6 | P6 | P1 | P1 | P5 | P5 | P1 | P1 |

b. What is the turnaround time of each process?

|  |  |  |
| --- | --- | --- |
| P1 | 120-0 | 120 units |
| P2 | 20−0 | 20 units |
| P3 | 40−20 | 20 units |
| P4 | 50−25 | 25 units |
| P5 | 110−45 | 65 units |
| P6 | 70−55 | 15 units |

c. What is the waiting time of each process?

|  |  |  |
| --- | --- | --- |
| P1 | 120-15 | 105 units |
| P2 | 20−20 | 0 units |
| P3 | 20−20 | 0 units |
| P4 | 25-20 | 5 units |
| P5 | 65-5 | 60 units |
| P6 | 15-15 | 0 units |

Consider the exponential average formula used to predict the length of the next CPU burst.

What are the implications of assigning the following values to the parameters used by the

algorithm?

Shortest Job First (SJF) is one of several CPU Scheduling strategies that aims to predict the length of the next CPU burst. The dynamic method describes two formulas for calculating SJF, mainly the simple average and exponential average methods. Using the exponential average method:

**Exponential average (aging)**

​ is the predicted value of the next CPU burst.

​ is the previous estimate of the CPU burst time.

​is actual length of CPU burst

is the weight given to the previous estimate

a. α = 0 and ԏ0 = 100 milliseconds

Where , the condition of the formula essentially does not apply.

- does not apply.

- , the actual length of CPU burst, becomes the predicted value of the next CPU burst (historical data not considered)

b. α = 0.99 and ԏ0 = 10 milliseconds

Where , the condition of the formula essentially does not apply.

- has a significantly reduced impact on the formula

- ​ is the previous estimate of the CPU burst time, has a significantly increased impact on the predicted value of the next CPU burst (historical data taken into account)