HOMEWORK 1

Anirudh Pal (pal5)

Answer 1: Isolation/Protection is achieved in a modern OS through the following mechanisms.

- **Hardware Features**
  - *Instruction Set:* The instruction set has two categories of instructions, which are privileged and unprivileged instructions. Unprivileged instruction can be executed by any process in the system. They perform operation that do not pose a risk to the security of the system. On the other hand, privileged instructions can pose a risk to security and are therefore only allowed to be used by processes with special permissions.
  - *Kernel/User Mode:* The above stated special permissions come in the form of kernel and user mode. These are hardware supported modes that allow only unprivileged instructions to executed when in user mode. When in kernel mode, both types of instruction can be executed. These modes also help limit accesses to some of the hardware like special registers.
  - *Trap/Untrap:* The trap and untrap instruction are part of the instruction set that allow switching between kernel and user mode. The trap instruction switches the CPU from user to kernel mode. The untrap instruction switches the CPU from kernel to user mode.
  - *Memory Protection:* Memory protection works on the concept of sandboxing processes. A processes is only allowed to read and write to its allocated location in memory. This isolates the processes to their own memory area and protects them from each other. This is usually implemented with paging or segmentation.
- **Software Features**
  - *Segregated Stacks:* In software we can allocate a runtime stack to each process (including the kernel). This works similarly to memory protection where the stack is only touched by its own owner and not some other malicious process.

The different mechanisms stated above stop both good and malicious programs from executing code outside their privilege level or access memory locations that belong to some other process, thus ensuring isolation/protection.

Answer 2: The read() system call requires access to memory locations (or devices) that are outside the scope of user process. Here is walkthrough of how it is executed while still maintaining isolation/protection.

- A user processes issues the read() function which is part of the API provided by the kernel.
- Trap Instruction is used to switch from user mode to kernel mode since the initial user process does not have the privileges to execute a privileged instruction.
- The stack also switches to the kernel stack to keep it isolated from the user stack.
- The instruction is carried out by the kernel (reactive) and a combination of privileged and unprivileged instructions are executed.
- As a result the required data is now available in the user process memory frame since it can only access its own frame.
- The untrap instruction switches the CPU back to user mode and the user process carries on execution as normal from its own stack and memory.

Answer 3: The following are hardware memory protection features that can be implemented is software without hardware support to achieve isolation/protection.

<table>
<tr><th colspan="2">Segmentation</th><th colspan="2">Paging</th></tr>
<tr><td colspan="2">In segmentation we have a table with bases, limits and privilege levels that are assigned during process creation. A process cannot access memory outside base and limit assigned to it.</td><td colspan="2">In paging a certain number of fixed size pages are assigned to a processes. The pages that are being used are kept in frames in RAM and rest are stored in secondary storage. A page table has to be used to resolve virtual addresses to physical addresses.</td></tr>
<tr><td>Pro</td><td>• Requires only a base + offset.
• Does not require secondary storage.
• Simple to implement.</td><td>Pro</td><td>• Can host more memory than the physical size of RAM.
• From processes perspective it has the entire memory.
• Tables can also be paged in and out based on need.</td></tr>
<tr><td>Con</td><td>• Variable size segment leads to external fragmentation.</td><td>Con</td><td>• Fixed size pages leads to internal fragmentation.</td></tr>
</table>

• The table has limited capacity and takes up memory. • Very slow with only software implementation.	• A clean cycle has to be performed to make more frames available in RAM. • Very slow with only software implementation.

Answer 4: Here is a review that presents the differences and similarities between Multics and modern OS.

Multics	Modern OS
Similarities	
• Adapt to hardware changes. • Supports virtual memory which stated as 2D memory. • Support multiple languages. • Multi-user. • Multi-tasking.	• Uses drivers to adapt to hardware changes. • Supports virtual memory through paging. • Also supports multiple languages. • Multi-user. • Multi-tasking.
Differences	
• I/O done only through interrupts and moving things between memory and device. • Variable segment size. • No clear differences between user and system processes. • Built-in mechanisms to backup data in secondary storage. • Small dedicated computers at terminals to handle some of the load on the main system. • A central Generalized I/O Controller is used for all the devices.	• I/O can be performed by DMA, i.e directly accessing the memory of device. • Fixed page size. • A clear distinction between user and system processes. • Generally doesn't have built-in backup mechanisms and require use of RAID or cloud storage. • All work is handled by the main computer. • Multiple different I/O interfaces available for different devices.