HW3 PART2
zw579 yz2455

1. What are the two requirements for successful completion of an application running in a Real-Time environment?

    a. Meeting individual timing requirement of each task.
    It is the time by which a system must produce an output. In hard real time system, if deadline is missed, it could be catastrophe.
    b. Correctness of output.
    The logical result of the computation must be correct.

    In a nutshell, the success of completing an application running depends on correct result within the time deadline.

2. Describe two processor innovations which, although they create a more efficient general purpose computing platform, would impact the correct evolution of applications in a real-time environment. Describe how these innovations operate and what aspects of the innovations interfere with real-time applications. For each of the innovations, how might they be adapted for use with a real-time system?

    a. DMA

    DMA is a technique by many peripheral devices to transfer data between the device and the main memory. The purpose of DMA is to relieve the central processing unit (CPU) of the task of controlling the input/output (I/O) transfer. Since both the CPU and the I/O device share the same bus, the CPU has to be blocked when the DMA device is performing a data transfer. Several different transfer methods exist.

    One of the most common methods is called *cycle stealing*, according to which the DMA device steals a CPU memory cycle in order to execute a data transfer. During the DMA operation, the I/O transfer and the CPU program execution run in parallel. However, if the CPU and the DMA device require a memory cycle at the same time, the bus is assigned to the DMA device and the CPU waits until the DMA cycle is completed. Using the cycle stealing method, there is no way of predicting how many times the CPU will have to wait for DMA during the execution of a task; hence the response time of a task cannot be precisely determined.

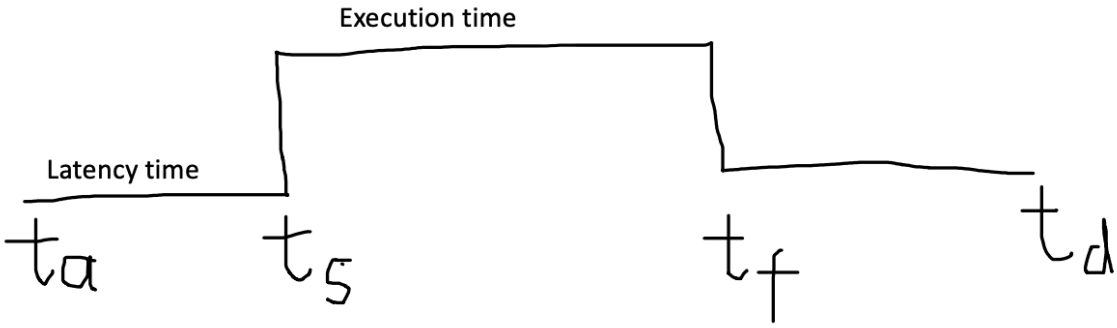
    To adapt real-time system, a different technique should be adopted. The memory time-slice method for DMA device is an option. According to this method, each memory cycle is split into two adjacent time slots: one reserved for the CPU and the other for the DMA device. Since the CPU and DMA device do not conflict, the response time of the tasks do not increase due to DMA operations and hence can be predicted with higher accuracy.

    b. INTERRUPTS

In order to get data from an I/O device, each task must enable the hardware to generate interrupts, wait for the interrupt, and read the data from a memory buffer shared with the driver. In many operating systems, interrupts are served using a fixed priority scheme, accord- ing to which each driver is scheduled based on a static priority, higher than process priorities. This assignment rule is motivated by the fact that interrupt handling routines usually deal with I/O devices that have real-time constraints, whereas most application programs do not. In the context of real-time systems, however, this assumption is certainly not valid, because a control process could be more urgent than an interrupt handling routine. Since, in general, it is very difficult to bound a priori the number of interrupts that a task may experience, the delay introduced by the interrupt mechanism on tasks' execution becomes unpredictable.

To adopt real-time system, one choice is to disable all external interrupts, except the one from the timer. In this case, all peripheral devices must be handled by the application tasks, which have direct access to the registers of the interfacing boards. Since no interrupt is generated, data transfer takes place through polling.

3. During investigation into embedded versus real-time systems, we have been discussing typical timing examples for processes using the parameters ta, arrival time, ts, start time, tf, finish time, and td, deadline time. Given this process timeline, what tools can we deploy to measure process execution time and process latency? Please include a timing diagram for a typical process, include the timing parameters discussed above, indicate process execution time and startup latency time on the diagram and discuss the tools used to measure each section.



We can directly use time tool to measure the execution time.

Cyclictest can measure latencies in real-time systems caused by the hardware, the firmware, and the operating system, it accurately and repeatedly measures the difference between a thread's intended wake-up time and the time at which it actually wakes up. Following is an example.

```
# cyclictest --mlockall --smp --priority=80 --interval=200 --
distance=0
```

And following is an example of result.

```
T: 0 (821)  P: 80  I: 200  C: 518063  Min: 1  Act: 1  Avg: 1  Max: 15
T: 1 (822)  P: 80  I: 200  C: 518050  Min: 1  Act: 2  Avg: 1  Max: 23
```

The parameters in the result show different things.

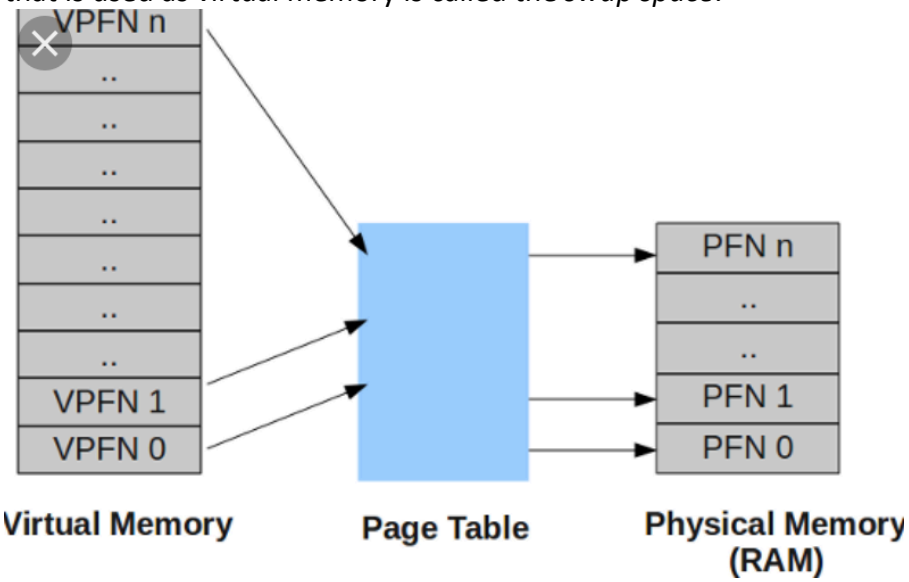
Abbreviation	Label	Description
T	Thread	Thread index and thread ID
P	Priority	RT thread priority
I	Interval	Intended wake up period for the latency measuring threads (in us)
C	Count	Number of times the latency was measured i.e. iteration count
Min	Minimum	Minimum latency that was measured (in us)
Act	Actual	Latency measured during the latest completed iteration (in us)
Max	Maximum	Maximum latency that was measured (in us)

4. Describe two of the four Linux kernel functions outlined in class. Figures could be helpful when describing these functions.

   a. Memory management

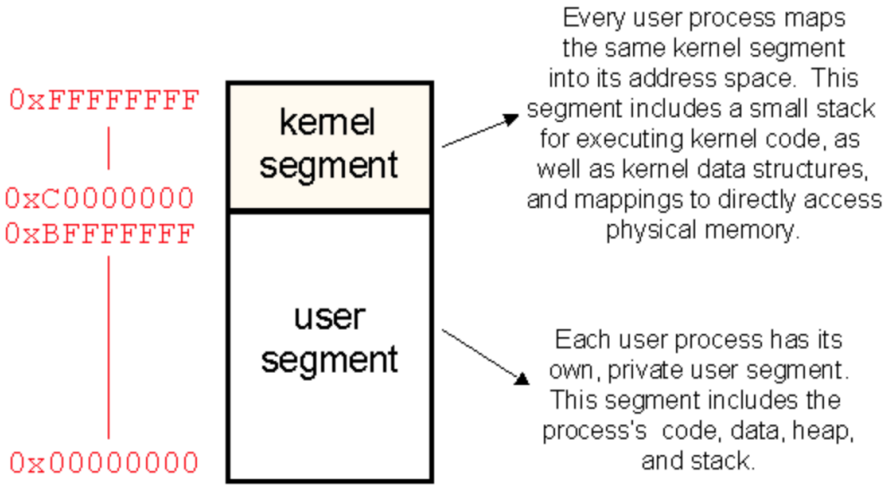
   Linux supports *virtual memory*, that is, using a disk as an extension of RAM so that the effective size of usable memory grows correspondingly. The kernel will write the contents of a currently unused block of memory to the hard disk so that the memory can be used for another purpose. When the original contents are needed again, they are read back into memory. This is all made completely transparent to the user; programs running under Linux only see the larger amount of memory available and don't notice that parts of them reside on the disk from time to time. Of course, reading and writing the hard disk is slower (on the order of a thousand times slower) than using real memory, so the programs don't run as fast. The part of the hard disk

that is used as virtual memory is called the *swap space*.



Linux can use either a normal file in the filesystem or a separate partition for swap space. A swap partition is faster, but it is easier to change the size of a swap file (there's no need to repartition the whole hard disk, and possibly install everything from scratch). When you know how much swap space you need, you should go for a swap partition, but if you are uncertain, you can use a swap file first, use the system for a while so that you can get a feel for how much swap you need, and then make a swap partition when you're confident about its size.

The x86 memory management architecture uses both segmentation and paging. Segmenting is possible to split the range of memory addresses that a process sees into multiple contiguous segments, and assign different protection modes to each. Paging is a technique for mapping small (usually 4KB) regions of a process's address space to chunks of real, physical memory. Paging thus controls how regions inside a segment are mapped onto physical RAM.

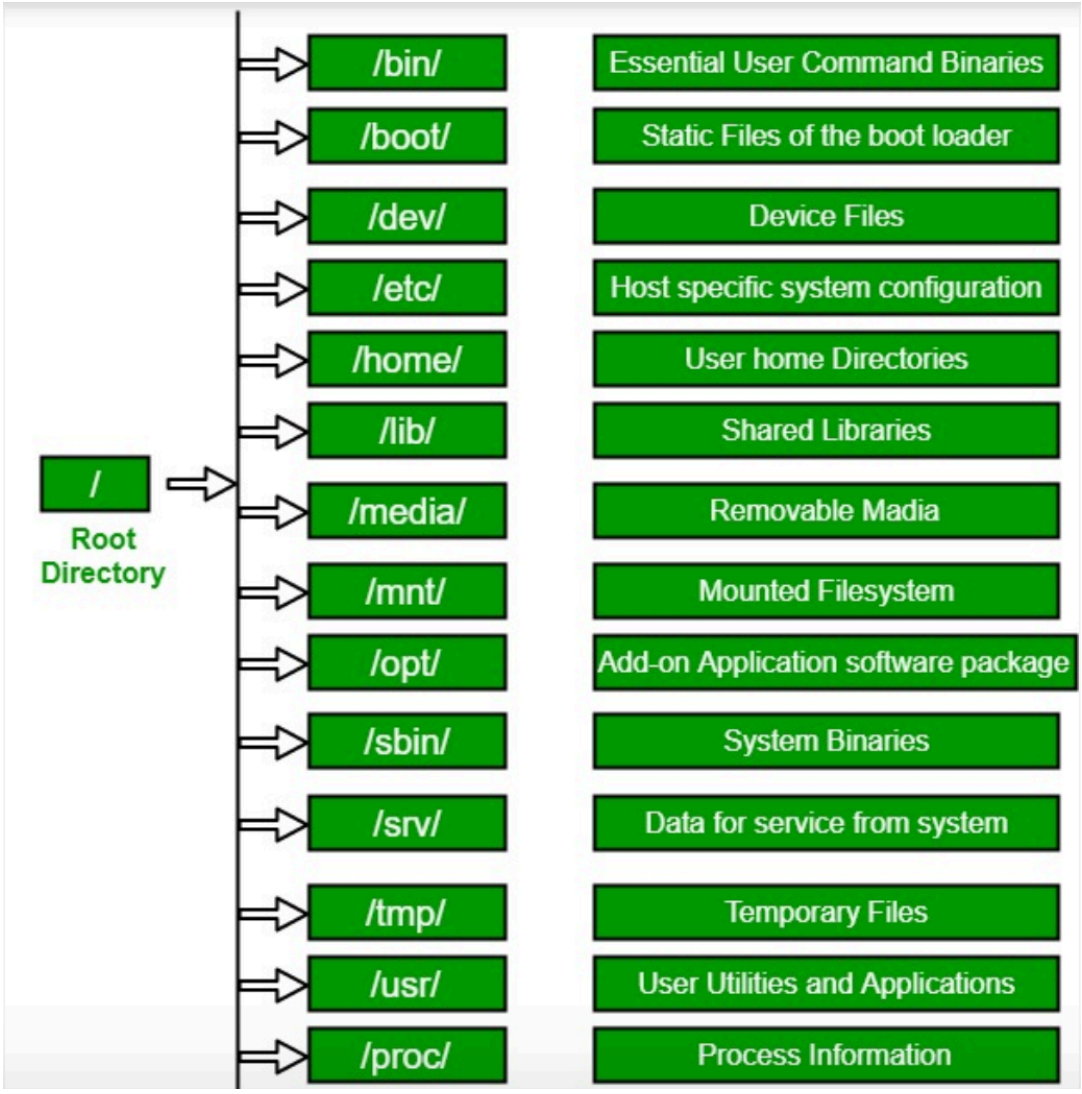


b. File system

File system in operating system is used to direct file sources. It stores the directory with tree structure.
The Linux File Hierarchy Structure or the Filesystem Hierarchy Standard (FHS) defines the directory structure and directory contents in Unix-like operating systems. It is maintained by the Linux Foundation.

- In the FHS, all files and directories appear under the root directory /, even if they are stored on different physical or virtual devices.
- Some of these directories only exist on a particular system if certain subsystems, such as the X Window System, are installed.
- Most of these directories exist in all UNIX operating systems and are generally used in much the same way; however, the descriptions here are those used specifically for the FHS, and are not considered authoritative for platforms other than Linux.

/ (Root) : Primary hierarchy root and root directory of the entire file system hierarchy. Every single file and directory starts from the root directory.



5. Describe the boot sequence of the Raspberry Pi in detail. What hardware modules are activated at each boot step? What software is used to boot the RPi? Where does each software process run? Identify the name and location of files used to boot the RPi. One of these files, start.elf, is an example of the ELF file format; What is the ELF format?

   Boot sequence:
   a. First stage bootloader: ARM GPU truns ON(CPU is off), GPU executes 1st stage bootloader from ROM
   b. 1st stage bootloader reads the 2nd stage bootloader "bootcode.bin" from SD card, loads it to L2 cache and runs it.
   c. 2nd stage bootloader enables SDRAM, loads the 3rd stage bootloader "loader.bin" to RAM and runs it. Srd stage bootloader reads the GPU firmware "start.elf".
   d. Start.elf reads system configuration parameters: config.txt, loads a kernel image "kernel.img", reads the kernel parameters "cmdline.txt" and release the reset signal on RAM.
   e. The kernel start booting.

   Elf is a kind of linkable and executable format, it is used to put right resource into memory and run it correctly.