COMP30640 UCD
Operating Systems 2016/2017

In-class Test
Introduction - Process Management (I and II) - Bash

Thursday 20th October, 2016

Instructions No document permitted. Use of Calculators prohibited.

Time Allowed 50 minutes.

Question 1. *Introduction* (40 marks)

1. Give three Operating System architectures and describe in your own words their characteristics.
   - Simple architecture: Only one or two levels of code. written to provide the most functionality in the least space; Not divided into modules: its interfaces and levels of functionality are not well separated
   - Monolithic architecture: every OS component is contained in the kernel and any component can directly communicate with any other (by means of direct function calls). This architecture tends to be highly efficient for this reason.
   - Layered architecture: Improves on monolithic kernel designs by adding structure: components are grouped into layers that perform similar functions; Each built on top of lower layers; Bottom layer (layer 0) is hardware; Highest layer (layer N) is the user interface. Advantages: modularity imposes structure and consistency (easier validation, debugging, modification, reuse). Problems with layered architectures: Appropriate definition of layers is difficult; a layer is implemented using only those operations provided by lower-lever layers; a real system structure is often more complex than the strict hierarchy required by layering; performance issues: processesâ requests might pass through many layers before completion (layer crossing) + system throughput can be lower than in monolithic kernels
   - Microkernel architecture: Minimises the services offered by the kernel, in an attempt to keep it small and scalable: Small core OS running at kernel level; OS Services built from many independent user-level processe; No consensus about minimal set of services inside micro- kernel (at least: minimal process and memory management capabilities, plus inter-process communications); Services such as networking and file system tend to run non-privileged at the user process level (i.e. out of the micro-kernel). Communication takes place between user modules using message passing. Benefits: easier to extend a microkernel; easier to port the operating system to new architectures; more reliable (less code is running in kernel mode); more secure. Drawbacks: performance overhead of user space to kernel space communication
   - Modular architecture: Many modern operating systems implement loadable kernel modules: Uses object-oriented approach; Each core component is separate; Each talks to the others over known interfaces; Each is loadable as needed within the kernel; Overall, similar to layers but with more flexible (e.g., Linux, Solaris).
2. Which operating system goals given below correspond to each of the following characteristics?
   - OS Goals:
     (a) Robustness
     (b) Scalability

(c) Security
(d) Portability
(e) Extensibility

- OS Characteristics
  (a) Users cannot access services of information without proper authorisation.
  (b) The operating system supports devices that were not available at the time of its design.
  (c) Hardware failure does not necessarily cause the system to fail.
  (d) The operating system runs on a variety of hardware configurations.
  (e) System performance increase steadily when additional memory and processors are added.

- Robustness: Hardware failure does not necessarily cause the system to fail.
- Scalability: System performance increase steadily when additional memory and processors are added.
- Security : Users cannot access services of information without proper authorisation.
- Portability: The operating system runs on a variety of hardware configurations.
- Extensibility: The operating system supports devices that were not available at the time of its design.

3. Explain briefly in your own words the following concepts: Time sharing Operating System and Real-time Operating System.
   A time-sharing OS allows many people to use a single system at the same time. Each user has the impression of being the sole user. This is done by dividing the processor time into small slots, and allocating those slots to users in a round robin fashion to set the response delay below that perceivable to humans. A real time operating system is a system that tries to guarantee the execution of processes within a specified time period. It does this through special scheduling algorithms. They are used in embedded appliances, guidance systems and other time sensitive applications.

4. Explain in your own words the following three concepts: multiprocessing, multiprogramming, multithreading.
   - Multiprocessing $\rightarrow$ Multiple CPUs
   - Multiprogramming $\rightarrow$ Multiple Jobs or Processes
   - Multithreading $\rightarrow$ Multiple threads per Process

Question 2. *Process and Threads* (40 marks)

1. What is the difference between a program and a process?
   Program is passive entity stored on disk (executable file), process is active. Program becomes process when executable file loaded into memory. A process is one instance of a program in execution A process requires a section of main memory to run. At any time, there may be more than one process running a different instance of the same program., e.g. several processes running the same editor. One program can spawn several processes.

2. Draw a diagram showing the life-cycle of a process.
   Figure 1 describes the life-cycle of a process

3. Are there any circumstances in which a transitions might occur between the states ready and waiting? Justify your answer.
   A process is ready when itâs waiting for the CPU, either after its creation, after an interrupt (the CPU has stopped it) or after being sent voluntarily to the waiting state (e.g., when requesting an I/O). Because the process is not in the CPU, nothing can happen to it (it is no running, hence its state cannot change) and it cannot be sent to a waiting state (which would require an event to happen to the process).

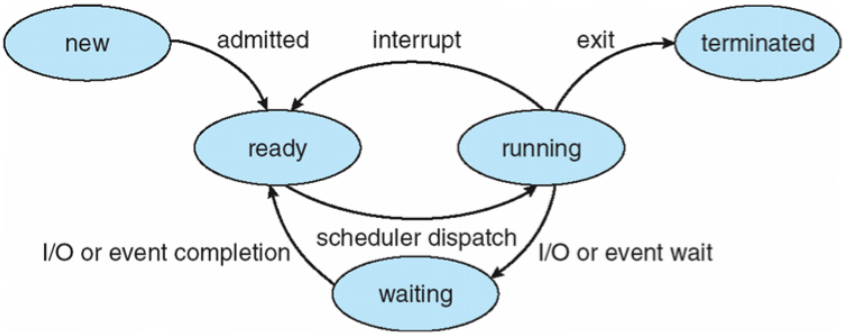

Figure 1: Life-cycle of a process (or thread).

4. Explain the concept of a context switch.
   Whenever the CPU starts executing a new process, the old process's state must be preserved. The context of a process is represented by its process control block. Switching the CPU to another process requires performing a state save of the current process and a state restore of a different process. This task is known as a context switch. When a context switch occurs, the kernel saves the context of the old process in its PCB and loads the saves context of the new process scheduled to run.

Question 3. *Bash* (20 marks)

1. What command do you use to obtain the absolute path of the current repository?
   pwd
2. Give a command that prints its parameter(s) on the standard output.
   echo
3. This is the structure of a condition in bash:

   ```
   if cond; then
   instructions
   fi
   ```

   And this is the structure of a while loop:

   ```
   while cond; do
   instructions
   done
   ```

   This is how conditions are written in bash:

   - Logical expressions on numerical values
     - `[ n1 -eq n2 ]`: true if n1 is equal to n2
     - `[ n1 -ne n2 ]`: true if n1 is different from n2
     - `[ n1 -gt n2 ]`: true if n1 is greater than n2
     - `[ n1 -ge n2 ]`: true if n1 is greater than or equal to n2
     - `[ n1 -lt n2 ]`: true if n1 is less than n2
     - `[ n1 -le n2 ]`: true if n1 is less than or equal to n2

   Logical expressions on string
     - `[ word1 = word2 ]`: true if word1 equals word2
     - `[ word1 != word2 ]`: true if is word1 different than word2
     - `[ -z word ]`: true if word is an empty word
     - `[ -n word ]`: true if word is not an empty word

Here's some variables related to commands' parameters:

- $0: the command's name
- $1 ... $9: the parameters
- $#: the number of parameters
- $@: list of the parameters
- shift: shift the list of parameters

Write a script `hello.sh` that greets every even parameters (2nd parameter, 4th parameter, 6th parameter, etc.):

```
$> ./hello.sh robert jane anthony saorla sean mary
hello jane
hello saorla
hello mary
```

The script follows these steps: while the number of parameters is greater than or equals 2, print the second parameter, and shift the parameters twice. (Another option could be to shift first, print the first parameter (the former second) and shift again).

```
#!/bin/bash

while [ $# -ge 2 ]; do
        echo hello $2
        shift
        shift
done
```