**Study Guide for Midterm Exam**

**Comp 4956 2015**

Lecture notes: Parts 1, 2, 3 and 4

Labs: Lab assignments and activities, and issues discussed in lab

Readings: as specified in assignments and power-point presentations

Project general architecture and your team’s solution and tasks

**Learning Outcomes / Questions**

1. What are the main two roles that the Windows registry has?

Administrative management and os management

1. What type of data is stored in the registry? Is this data static or dynamic?

Registry are data structures that are configurations and controls for the operating system. They are data static

1. Where the registry is physically stored?

Stored in the hard disk

1. Is the registry part of the memory pool? Why or why not?

Yes. Registrys are interfaces to several of in-memory structures maintained by the executive and kernel

1. Give 3 examples of data types defined for the registry.

Reg\_sz: string used to hold paths and names etc

Reg\_dword: hold numbers and booleans

Reg\_binary: holds numbers over 32 bit and raw data such as encrypted passwords

1. Describe the six root key of the registry. Which key is not visible in the registry’s editor?

The security key (this includes sam as well since sam is linked to it) is not visible which is located in hkey local machine

Hkey\_local\_machine

Hardware

Software

System

Sam (security access mananger)

security

Hkey\_user

Hkey\_current\_user

Hkey\_performance\_data

Hkey\_classes\_root

Hkey\_current\_config

1. Per-user class registration was introduced first in Windows 2000. Why per-user class registration has been added to the registry?

To separate per-user class registration from system-wide class registration. Non-privileged users can delete modify keys without affecting the overall system

1. Regarding the registry, where is the performance data stored? Describe the mechanism for accessing the performance data.

Performance data are not stored in the hkey\_performance\_data. The key is used to get information on the performance data from the performance providers

1. What is the Windows Management Interface?

WMI is a management interface implemented to remove the limitations of the older management tools. Used to determine the machines performance and benchmark

1. In what language(s) is (are) the WMI written?

Not limited to a single language: can be used in win32, jscript, python, c#, vb, etc.

1. What are the WMI features?

Supplies management to other areas in the os, scom, and winrm

Scom – security center operation manager. Data center management for the os. Places an agent (piece of software) onto the computer to be monitored

Winrm – management protocol. Soap (simple object access protocol)

Wmi sdk for download and contains documentations

Programmers can write scripts or applications to perform tasks such as automating administrative actions, get performance data, etc

1. Why is needed a common WMI?
2. What are the WMI advantages?

Works locally and remotely

Bi-directional

Extensible

Windows object can be observed

Extensible to other objects

1. How is WMI implemented in the .NET framework? What are the advantages and limitations of a .NET development platform for WMI?

Implemented using classes in system.management.instrumentation.

advantages

Brings script-like simplicity

Development of application and providers is quicker.

Easy debugging

Access to all wmi data

Limitations

Cannot define methods

Cannot expose writeable properties on new classes that are not wrappers of unmanaged wmi clases

Can only use embedded objects when defining new classes

1. In your assignment, what did you use to access the disk/process data? What about the registry?

Management class and objects; win32\_logicaldisk for disk and win32\_process for process data. Microsoft.win32.registrykey for registry.

1. Describe in your own words the process creation and termination in UNIX, Linux and Windows.

Unix/linux:

-fork() function invoked by system

-allocate slot in process table for new process

-assign unique id

-make copy of process image of parent

-increment counters for any files owned by the parent

-assign process state to ready to run

-return id to parent process and 0 to process

Windows:

-assign unique id

-allocate space for process

-initialize pcb

-set appropriate linkage

-create/expand other data structures

Termination:

-os frees up resources used by the process

1. What is happening when the system call fork() is called in UNIX or Linux?

Process creation (see 16.)

1. What is the role of the instruction wait() in: child\_pid = wait(&status);

Stop current process until child process is finished and changes states

1. What is a thread? Describe a thread data structure and characteristics in details. How is a thread different from a process and a fiber in Windows OS? What is the equivalent of a fiber in UNIX?

Threads are contained by processes.

Each process must have at least one thread (called primary thread).

Processes are created from user level but creation is executed in kernel.

Threads are unit of execution.

Threads can create fibers, similar to threads but scheduled by the thread.

1. What is the difference between the microkernel and the Executive in Windows OS (what are their roles)?
2. How do processes communicate? (Consider all IPC mechanisms and shared memory method). What about threads?
3. How many stacks are implemented for threads in Windows? Why?
4. What is the role and details of implementation for HAL.dll? What is its content?
5. What is NTOSKRLN. EXE?
6. What are the process’ components in Windows? What about thread’s components?
7. Describe issues of vulnerability related to execution in kernel mode.
8. Discuss kernel-mode vs. user mode.
9. Draw the diagram of Windows OS indicating the user and kernel mode.
10. Name 5 .dlls that represent important parts of the Windows OS.
11. Explain the multithreaded execution in Windows (switching from one thread to another).
12. What is the benefit of using \_beginthreadex()?
13. Explain why the thread executions in assignment Multithread.exe (written in C) do not end at the same value of the counter.
14. Explain why the thread executions in assignment Multithread.exe (written in C#) end at the same value of the counter.
15. How many threads were created in your .NET Multithread assignment? Why?
16. What was the role of a delegate in the .NET Multithread assignment?
17. How is a semaphore implemented? Write the pseudo-code of the functions signal(), wait() and init().
18. Describe the functionality of a semaphore by giving an example of a situation that is solved with semaphores.
19. Describe the functionality of a mutex by giving an example of a situation that is solved with mutexes.
20. What is a monitor?
21. What is the difference between a monitor and a semaphore?
22. What situations are better solved with monitors? What about semaphores?
23. Write the pseudo code for the producer – consumer with a circular buffer for the most general situation which will satisfy multiple producers and multiple consumers:
    1. With semaphores
    2. With monitores
24. How are semaphores / monitors implemented in Windows OS and .NET?
25. Consider the multithread assignment:
    1. What solution did you use in the multithreaded assignment to communicate between the counting threads and the timer thread? Why?
    2. What safety mechanism should be implemented for their communication?
26. Explain the differences between beginthreadex() and CreateThread().
27. In your assignments you used Executive objects exposed via the Win32 interface or the .NET framework.
    1. Name four of them.
    2. Explain the Executive access to an object. How does an Executive component specify the use of an object?
28. In the .NET framework you have available several primitives for synchronization. Compare and contrast the Monitor, Semaphore and Mutex classes. For each explain what kind of synchronization is used: at the level of Kernel Windows objects or not. In what circumstances will you decide to use one of them?
29. What is the base and dynamic priority of threads in Windows? Why do threads have dynamic priorities?
30. In the Multithread assignment: Is the number of context switches performed by the dispatcher for each thread equal? Why?
31. In the Multithread assignment: What is a thread doing in kernel mode?