操作系统原理		September 13, 2011
	Homework 2	
计算机 92 班		戴唯思 09055029

2.1

> The services and functions provided by an operating system can be divided into two main categories. Briefly describe the two categories and discuss how they differ.

Operating systems provide services which directly *translates* operations from processes to hardware, and ones not directly linked to external devices.

The first sort of services acts as the fundamental of implemenation of executables, and the others are designed to make it convenient to program and to equip applications with similar environment(e.g. POSIX). Within the operating system, the latter ones are usually realized on the basis of the former ones.

2.2

> List five services provided by an operating system that are designed to make it more convenient for users to use the computer system. In what cases it would be impossible for user-level programs to provide these services? Explain.

1. File systems
   File systems are a low-level database management system provided by the operating system in order to organize data. Privileges to maintaining file systems, a systemwide key-value data lookup function should be restricted to the operating system itself, for user-level programs can mess it all up.
2. Direct hardware access
   The operating system should be the only role to access hardware, or pass operations to hardware. Any applications requesting straight entry to devices should be examined by the OS.
3. Networking
   Networkings are an approach to generic data exchange. In modern computer system structure, networking should be considered systemwide and thus be provided by the OS.
4. Data execution detect
   The OS can check if specific code should be runnable, to reduce the possibility of data damage.
5. Program execution
   The OS functions as a process *controller*, decides which part of codes are to be loaded.

2.3

> Describe three general methods for passing parameters to the operating system.

1. Pass parameters in *registers*,
2. store parameters in a *table* in the memory, and pass the *address of the first block*,
3. push parameters to a *stack*, changes context and later pop it out.

2.7

> What is the purpose of the command interpreter? Why is it usually separate from the kernel? Would it be possible for the user to develop a new command interpreter using the system-call interface provided by the operating system?

Command interpreters tries to understand and execute commands which are entered interactively by a human or a program.

Shells, such as bash which I use everyday, and ~~Command Prompt~~ in Windows which can be omitted when compared to a ordinary Unix shell, are implemented using the system-call interface provided by the operating system. So a capable user can absolutely develop one similarly.

2.8

> What are the two models of interprocess communication? What are the strengths and weaknesses of the two approaches?

- Message passing
  - strengths: program structures better separated, dangerous operations firewalled
  - weaknesses: message passes more slowly, for symmetrical copy operations are to be made
- Shared memory
  - strengths: fast and direct
  - weaknesses: unexpected behavior when unauthentic programs wrongly accesses the shared memory segments

References

1. https://secure.wikimedia.org/wikipedia/en/wiki/Operating_system
2. http://support.microsoft.com/kb/875352
3. http://www.personal.kent.edu/~rmuhamma/OpSystems/Myos/sysService.htm
4. http://searchwinit.techtarget.com/definition/command-interpreter
5. http://stackoverflow.com/questions/573623/powershell-vs-unix-shells
6. http://en.wikipedia.org/wiki/Concurrent_computing
7. http://armstrongonsoftware.blogspot.com/2006/09/why-i-dont-like-shared-memory.html
8. http://homepages.cae.wisc.edu/~mikko/757/sect1.pdf