Tina Peng

CMPS111 Harrison

13 April 2018

Homework 1

Question 1. Briefly outline the evolution of Operating Systems from those used by the earliest stored program computers of the 1940s to their modern counterparts.

The operating systems started as batch job control systems, where human operators manually set up runtime context for each program and operating systems worked through individual jobs in a batch and a linear fashion. Then came the spooling batch system which required hardware interrupts and processed jobs (similar to CPU instruction pipelining). Soon later came multi-programmed batching, where multiple jobs to be in memory at the same time and share a single CPU. This created more sophisticated operating systems, introducing ideas of job scheduling, memory management, and CPU scheduling.

Then in 1956, the 7030 pioneered interrupts, memory protection, multiprogramming, and many other advance computing concepts. And naturally evolved from multi-processed systems came time sharing systems, where there was now user interaction and multiple users can share the same machine. Eventually the first supercomputer was made in 1964 and introduced the simultaneous processing OS, although it unfortunately failed.

Seymour Cray then came along in 1969 and helped make the CDC 7600 with the first functional timesharing OS. Then he created the Cray-1 in 1976 that had the Cray Time Sharing System.

Soon afterwards, there was a paradigm shift to create personal computers that were more user friendly and cheap. Nowadays, modern computers can provide multiple processing units in a shared memory allocate.

Question 2. In the following piece of C code, how many processes are created when it is executed? Explain your answer. int main() { fork(); fork(); fork(); exit(1); }

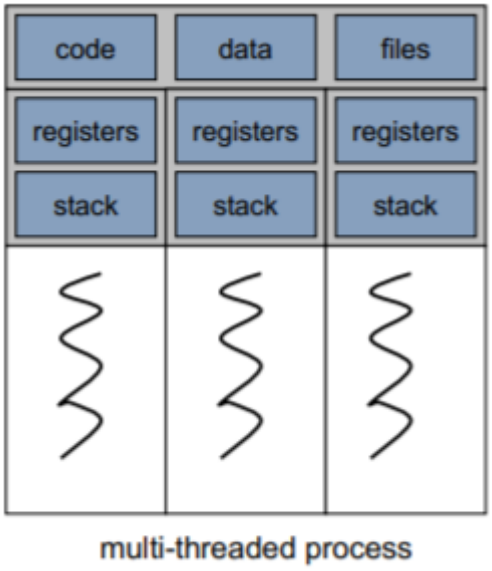
fork() is a function that duplicates the parent and returns a child process. Therefore, each fork() creates a new process (while the parent gets exited), therefore 3 processes get created when the program is executed. If we include the fact that starting a program also creates a process, the total number of processes created is 4.

Question 3. If an Operating System assigns an unsigned 32bit integer to store current time as the number of seconds elapsed since 00:00 on January 1 1970, is this likely to be a problem? Explain your answer.

As of right now, it is not a problem because a 32bit integer can still hold the amount of seconds needed to represent any time in the year of 2018. A 32bit integer can at max, represet 2^{31}-1 = 2,147,483,647 seconds. However, any number higher will result in an integer overflow, causing the date to be misrepresented. For example, if the date continued onto the year January 19, 2038, the seconds needed to represent the amount of seconds will be too great, causing a problem (the overflow will end up misrepresenting the date to be on December 13, 1901).

Question 4. Describe how a web server might leverage multi-threading to improve performance. Include diagrams if you feel this will make your answer clearer.

Multi-threading improves performance by splitting up the work and allowing multiple CPUs to work on different tasks at the same time, combining the results at the end. This improves performance when certain tasks (the ones with no dependencies) can be processed simultaneously, instead of unnecessarily waiting for the computer to complete other tasks. On a web server, the threads can be split for each HTTP request, one for rendering, and one to take in user input when needed. This allows pages to be rendered and accounts for user input at the same time.



multi-threaded process

Here is a diagram from the notes that shows how a thread process can be split into multiple threads.

Question 5. (a) In a multiprogrammed environment with 16MB of memory where all processes require 1MB of unshared memory and spend 60% of their time in I/O wait, calculate how much memory will remain unused when approximately 99% CPU utilization is achieved.

The CPU utilization formula is $1 - p^n$. Since 99% CPU utilization is achieved, the formula can be set to $1 - p^n = .99$. So p= the percent of time waiting for I/O and n = the number of processes so the formula can then be substituted into $1 - (.6)^n = .99 \rightarrow (.6)^n = .01$. Taking the log on both sides would reveal that n = 9.01515 and therefore 9 processes (10 if we round up) have been used. Each process requires 1MB of unshared memory so 9.01515/1 = 9.01515. Since there are a total of 16MB memory, 16 – 9.01515 = about 6.984MB that remains unused.

(b) In the same multiprogrammed environment, if each process now requires 3MB of unshared memory, calculate the maximum achievable CPU utilization. Show all your work.

The CPU utilization formula remains to be $1 - p^n$. If there are 16MB of memory and each process requires 3MB of unshared memory, then there can be a max of 5.33 aka 5 processes (we must round down). Therefore, the maximum achievable CPU utilization is $1 - (.6)^5 = .92\%$.