COSC 6360: Operating Systems
PAPERS ON THE FIRST FALL 2020 QUIZ

You are only responsible for the materials discussed in class as they are ***summarized*** in the handouts and discussed in the PowerPoint presentations. I expect you to understand these summaries and to be able to comment around them.

Always ask yourself ***why*** a specific technique was used and ***which*** problem it tried to solve.

UNIX

D. M. Ritchie and K. Thompson, "The UNIX time-sharing system," *The Bell System Technical Journal,* 57:6 (1978), pp. 1905-1929.

M. K. McKusick, W. N. Joy, S. J. Leffler and R. S. Fabry, "A Fast File System for UNIX," *ACM Transactions on Computer System*s, 2:3 (1984), 181-197.

J. S. Quarterman, A. Silberschatz, and J. L. Peterson, "4.2 BSD and 4.3 BSD as examples of the UNIX system," *ACM Computing Surveys,* 17:4 (1985), pp. 379-418.

The handout is self-contained but lacks diagrams: consult also the PowerPoint presentation. You might skip the section on sockets but should pay particular attention to process creation, the Fast File System and the BSD page replacement policies.

Virtual Memory

O. Babaoglu and W. Joy, "Converting a swap-based system to do paging in an architecture lacking page-reference bits," *Proc. 8th ACM Symposium on Operating Systems Principles*, (1981), pp. 78-86.

Focus on:

- *How Berkeley UNIX simulates the page-referenced bit in software. (We did not discuss why Babaoglu and Joy rejected other page replacement policies.)*
- *What they did to limit the context switch overhead.*
- *Why they introduced the **vfork()** system call.*
- *How the page replacement policy evolved since the early eighties and why.*

Start with the handout and consult the paper when you don't understand a specific paragraph.

Review problems

1. How many lines will the following program print?

```
main() {
      fork();
      printf("Hello!\n");
      fork();
      printf("Goodbye!\n");
} // main
```

2. How does copy-on-write reduce the cost of `fork()` system calls?

3. Why are ***metadata updates*** much ***costlier*** than data updates?

4. What is happening when the hand of the Berkeley Clock page replacement policy moves ***too fast***?

5. Consider the ***Two-Handed Clock*** policy of Berkeley UNIX, where the page-referenced bit is simulated by software.

a) What would happen if the second hand follows ***too closely*** the first one?

b) What would happen if the second hand is ***too far apart*** from the first one?

6. Unlike the older UNIX file system, the Fast File System specified a minimum block size.

a) What was that ***minimum block size***?

b) What is the ***main advantage*** of that ***specific value***?

Solutions 1. VI. 2. Copy-on-write allows parent and child processes to share the same address space until the child process performs an exec() by selectively copying the pages that one of them modifies. 3. They have to be done in a well-defined order and the "Fast" file system uses blocking writes to implement them. 4. "Too many page frames get invalidated, which results into too many calls to the page fault handler to revalidate them. 5. a) Some pages would be expelled too quickly. b) Pages would remain too long in main memory after their last access. 6. a) Four kilobytes. b) It is the minimum block size that allows access to four gigabytes of data with two levels of indirection.