CIS 657
HOME WORK 1

Please type your answers and submit your solution on blackboard. Work is to be completed individually!

(1) On all modern computers, at least part of the interrupt handers are written in assembly language. Why?
(2) If a multi-threaded process forks, a problem occurs if the child gets copies of all the parents threads. Suppose that one of the original threads was waiting for keyboard input. Now two threads are writing for keyboard input, on in each process. Does this problem ever occur in single-threaded processes?
(3) Why would a user thread ever voluntarily give up the CPU by calling thread yield? After all, since there is no periodic clock interrupt, it may never get the CPU back.
(4) Consider a process that continuously reads in a chunk of data from disk (1MB say), does a heavy computation over that 1MB and then reads in the next chunk (ad infinitum). How would this Multi-Level Feedback queues handle this process?
(5) In the three following situations explain which model we should use:
    (a) user-level threading
    (b) kernel-level threading
    (c) hybrid-threading
    (d) no-threading (i.e. single-threaded application)

    Explain clearly what the advantages are to the choices you have made, or why the other options are not viable. While making your decision consider things like the amount/type of I/O your application requires.

    (a) You are writing an application for an operating system (that support threading). The application needs to continuously and simultaneously fetch information over a network connection, accept input from the user, and format the display to the user.

    (b) You are writing an application for a single-threaded operating system (i.e. the OS does not support threading). The application waits for a packet of internet information, and then processes the information. The packets come once an hour, and processing takes only a few minutes.

    (c) You are writing an application to monitor a realtime system (i.e. a manufacturing floor). The OS supports threads but not 'real-time' threads (the o.s. has no way to fulfill specific timing requests). The application needs to continuously monitor aspects of the real-time system it is monitoring with specific

requirements as to the timing. (i.e. every 5 seconds check the heat gauge). If there are N aspects of the real-time system that need to be monitored, each 'check' of that aspect may call a blocking system call.

(6) Discuss why the solution in which every philosopher takes and locks the fork on there left and then tries to grab the fork on their right fails.

(7) For the following synchronization example answer all parts. Santa Claus sleeps in his shop at the North Pole and can only be awakened by either (1) all nine reindeer being back from their vacation, or (2) the elves are having difficulty making toys. There are 3 types of threads: santa claus (1), reindeer (9), elves (N). Constraints:

- The last reindeer to arrive must get Santa while the others wait in a warming hut before being harnessed to the sleigh.
- After the ninth reindeer arrives, Santa must invoke prepareSleigh, and then all nine reindeer must invoke getHitched.
- To allow Santa to get some sleep, the elves can only wake him when three of them have problems.
- When three elves are having their problems solved, any other elves wishing to visit Santa must wait for those elves to return.
- After the third elf arrives, Santa must invoke helpElves. Concurrently, all three elves should invoke getHelp.
- All three elves must invoke getHelp before any additional elves enter (increment the elf counter).
- Santa should run in a loop so he can help many sets of elves.

a) Describe the purpose for each of the global variables below:
i) reindeerSem:
ii) santaSem:
iii) mutex:
iv) reindeer:

b) Write the elf portion of the synchronization routine.

(8) Given a 48-bit addressable CPU, a page size of 8KB, a page entry size of 8 bytes, and a ram size of 4GB answer the following questions. You can leave your answers in the form 2x (where you would give x).
a) How big is the (maximum) virtual address space?
b) How many entries are there in the page table (assuming the maximum virtual address space size)?
c) How large (in bytes) would the page table be?
d) How large (in bytes) would an inverted page table be?

```
Semaphore santaSem = new Semaphore(0);
Semaphore reindeerSem = new Semaphore(0);
Semaphore elfTex = new Semaphore(1);
Semaphore mutex = new Semaphore(1);
int elves = 0;
int reindeer = 0;
void santa(){
   while(true) {
        santaSem.P();
        mutex.P();
        if (reindeer == 9) {
                prepareSleigh();
                for(int i = 0; i < 9; i++)
                        reindeerSem.V();
        }
        else (if elves == 3) {
                helpElves();
        }
        mutex.V();
   }
}

void reindeer()
{
        mutex.P();
        reindeer += 1;
        if (reindeer == 9) {
                santaSem.V();
        }
        mutex.V();
        reindeerSem.P();
        getHitched();
}
```

FIGURE 1

e) Using a two level multi-level page table where each second- level page table would be a page-size large, what would an address look like? (i.e. How many bits for offset, 2nd level and top level fields?)

f) Explain how translation from a virtual address to a physical ad- dress occurs using a multi-level page table.

(9) Given 4 page frames which are all initially empty state give the number of page faults caused by FIFO, Optimal page replacement and LRU (assume no periodic clock clearing of R-bits occurs) given the memory references below. Also give the

final contents of the 4 page frames.
1) read at page 0
2) write at page 2
3) read page 3
4) write page 4
5) read page 5
6) write page 3
7) read page 0
8) write page 2
a) FIFO: Number of page faults: Final contents:
b) Optimal:
Number of page faults: Final contents:
c) LRU:
Number of page faults: Final contents:

(10) I-Node System Given that the block size is 32 bytes, and a disk address is 4 bytes, the RAM is 1MB and the i-node structure is as depicted answer the following questions:

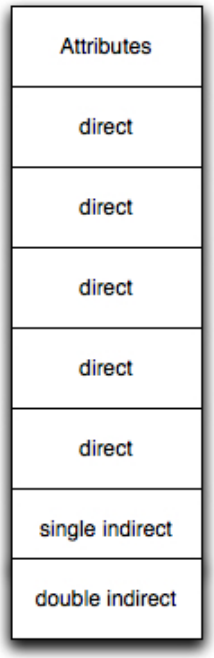

FIGURE 2

a) How many addresses fit in an i-node?
b) (3) What is the maximum file-size?

c) (5) How many (maximally-sized) files can fit on the disk?
d) (5) Use the example in Figure 2 to convert the following file offsets (in bytes) to physical block addresses:
100, 301, 633

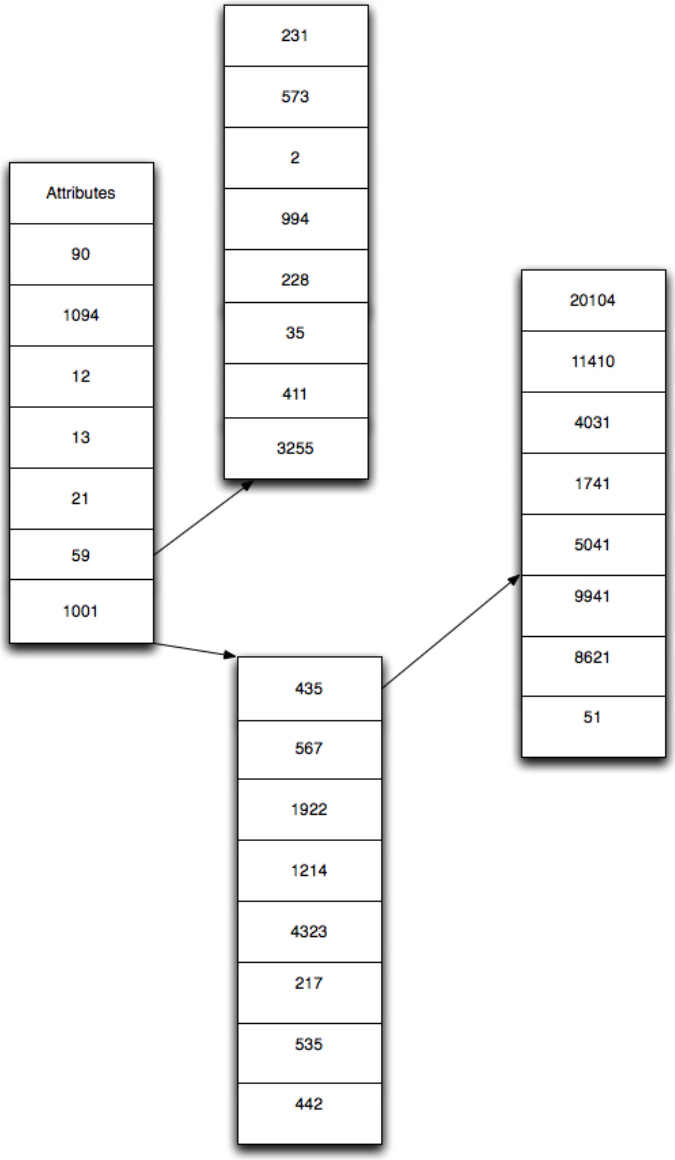

FIGURE 3