

THE UNIVERSITY OF HONG KONG

FACULTY OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND INFORMATION SYSTEMS

CSIS0230A Principles of Operating Systems

Date: December 19, 2003

Time: 9:30 am–12:30 pm

Candidates may use any calculator which fulfils the following criteria: (a) it should be self-contained, silent, battery-operated and pocket-sized; (b) it should have numeral-display facilities only and should be used only for the purpose of calculation; (c) it should not have any printing device, alphanumeric keyboard, or graphic display; and (d) it should not contain any recorded data or program. It is the candidate's responsibility to ensure that the calculator operates satisfactorily and the candidate must record the name and type of the calculator on the front page of the examination scripts. Lists of permitted/prohibited calculators will not be made available to candidates for reference, and the onus will be on the candidate to ensure that the calculator used will not be in violation of the criteria listed above.

This is an "open-book" examination. Be reminded that, in your answer, you may refer to materials of the course including (1) the textbook and reference book of the course, (2) sample solutions of any assignment and quiz, and (3) reading materials, worksheet and solutions of tutorials. If you do so, please clearly list the source of your reference together with the section, page and line number. You don't need to copy materials verbatim to the answer book.

Answer all questions in the answer book provided. Each of the 7 questions is worth 10%, and the marks of the 3 questions you score the most will be doubled.

1. (*Process control*)

- a. (5%) What is the output of the following C program when running on the terminal, assuming that all the *fork()* succeeds? When each line is printed?

```
/* #include's omitted for brevity */
int main() {
    int i;
    for (i = 0; i < 2; ++i) {
        int is_parent = 1;
        if (fork() == 0)
            is_parent = 0;
        sleep(i+is_parent);
        wait(0);
        printf("i=%d, is_parent=%d\n", i, is_parent);
    }
    return 0;
}
```

- b. (5%) Write additional code to the above program, so that if Control-C is pressed on the terminal during the execution (thus generating SIGINT), exactly one message "I am interrupted" is printed, and after that all processes of the program should be terminated by the SIGINT signal. Your modifications should not change the behaviour of the original program, and should not create additional processes.

2. (*Virtual Memory*) In the frame table that an operating system kernel uses to keep track of the sharing of frames among processes, there is a location that stores a “usage count”.
- (4%) Is it possible that the count is stored within the page table rather than the frame table? If so, how it will be accessed? If not, why?
 - (3%) When a new program is compiled and executed, we expect that the code segment of the program is not shared with other programs. But in reality, the usage count of such frames is more than 1. Who is the other user of the frame? With this user, when will the frame count become zero and get deallocated?
 - (3%) It is a common technique for the kernel to keep a usage count for deallocation of resources. Give an example on (i) the filesystem interface, and (ii) the filesystem implementation.
3. (*Address resolution*) The following C code is compiled into position independent object code using the `-fPIC` flag of `gcc`, which is then used to produce a shared library.

```
struct int_node_t {
    int data;
    struct int_node_t *next;
};
static struct int_node_t start_node = { 0, NULL };
static struct int_node_t *int_list = &start_node;
struct int_node_t *search_elt(int n) {
    struct int_node_t *curr;
    for (curr = int_list; curr != NULL; curr = curr->next)
        if (n == (int) sqrt(curr->data))
            return curr;
    return NULL;
}
```

- (3%) The `search_elt()` function is called with an argument `n`. Is a relocation needed to find the address of `n`? Why?
 - (3%) The `search_elt()` function calls the `sqrt()` function in another shared library. Is it possible that the address of `sqrt()` to be resolved when (1) the shared library is produced by linking, (2) the shared library is loaded, and (3) the function is called? Why?
 - (4%) Explain why the declaration of `int_list` causes a relocation entry to be put in the shared library. What is the best time when the relocation is done? Why?
4. (*Process synchronization*) When there are multiple processors, one way to establish mutual exclusion within the Linux kernel is to use **spin-locks**.
- (2%) Explain what is a spin lock, and how it is typically implemented.
 - (4%) Explain why it is undesirable for the system to switch context when the current process is holding a spin lock.
 - (4%) When holding a spin lock, you should not call `copy_from_user()`. Explain the reason, using a scenario in which this drastically reduces the efficiency of the system.

5. (*Deadlocks*) One way to deal with deadlocks on the use of resources is to ask every process to declare a maximum requirement. When processes request for resources, the system uses the banker's algorithm to check whether the system is deadlock safe, and allocate the resource only when it is safe.
- (4%) Suppose a new process (currently holding no resource) increases its maximum requirements. Explain why the system cannot become unsafe by this operation.
 - (4%) An environment involves 2 resources A and B, and 4 processes 1, 2, 3 and 4. Process 1 tries to allocate 1 copy of A. When the banker's algorithm is executed, it is found processes 1, 3 and 4 remains. So process 1 is put to wait. The next operation on the resources is done by process 4, which reduces the maximum requirement on resource B. Can this lead to immediate allocation of resource A? If not, explain why. If so, give all details (allocation and maximums of all processes together with the total number of available resource) of one situation that process 1 indeed obtain the requested resource immediately after the event.
 - (2%) To interact with another part of the program, it must also use mutexes and conditions, not represented by the resources in the banker's algorithm. Suggest a way to coordinate the use of the mutexes and the banker's algorithm to prevent deadlocks.
6. (*Scheduling*) A Linux user has a workstation which runs some interactive programs. He needs to run a background program that performs a lot of disk I/O for a long period of time. It is adversely affecting the behaviour of interactive programs when they also use the disk. Having heard about the `nice` command, he uses it to run the background program: the interactive programs run at nice level 0, while the tasks performing disk I/O run at nice level 19. No program is consuming a lot of CPU time in the system.
- (4%) Explain why it is unlikely that `nice` is effective in restoring the responsiveness of the interactive tasks.
 - (3%) It is found that, counter-intuitively, adding a CPU-intensive task at nice level 5 would actually improve the situation. Explain how this happen.
 - (3%) Suggest a way to get the same effect without increasing the CPU load.
7. (*Filesystem implementation*)
- (5%) The inode structure of a modern Unix filesystem is usually the limiting factor of the size of files in it. What is the maximum possible file size for an ext2 filesystem with 2KiB blocks? Show all your reasoning.
 - (5%) Suppose you have an ext2 filesystem that cannot be mounted due to filesystem consistency problems. You have a tool which allows you to look at inodes and data blocks directly. Explain how you will find the data of a particular file `/boot/bzImage` in the filesystem, assuming that all the needed structures to it are not corrupted. Explain all the needed steps.

END OF PAPER