

### **Multiple choice questions: (40 pts)**

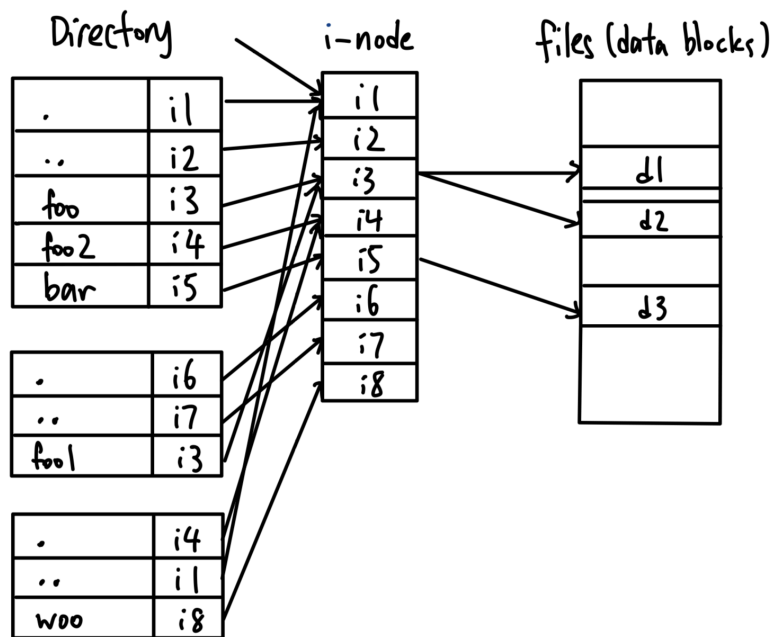
5pts each, deduct 3pts if one answer is wrong, the lowest score for each question is 0pts (Put down the answers - explanation is not required)

1. Which of the following about Unix is/are true?
  - a. The first Unix Operating System was built by Linus Trovalds
  - b. The PID of the shell process in a Unix system is usually 0
  - c. In Unix, users often interact with the OS kernel via a special program called a shell. One can choose the shell that she wishes to use by default modifying the environment variable
  - d. POSIX standardizes the services, including the system calls that a Unix system must provide
  - e. In Unix, system calls generally take much longer than function calls to finish
  
2. Which of the following is/are true about a process?
  - a. A process created by `fork()` always starts running immediately
  - b. A process created by `fork()` inherits all file descriptors from its parent process
  - c. `vfork()` shortens the process creation time because the kernel employs copy-on-write to avoid the memory copying overhead
  - d. A process that `exec()` a program can never become an orphan process
  - e. In Unix, whenever a process terminates, the OS kernel recycles all of the process' respective resources, so the parent of the terminating process does not have to perform extra recycling work
  
3. Assuming that you log in to the linux1 server with your account, you can read or execute the file "hello" from the server. You are in a group created for all students (GID = 7), while all faculty members are in another group (GID = 8). Which of the following is/are true about the file?
  - a. The UID of the file is the root user. The access permission is set to 4700
  - b. The UID of the file is the root user. The GID of the file is 3. The access permission is set to 4750
  - c. The UID of the file is you. The access permission is set to 555
  - d. The UID of the file is another student who takes SP. The GID of the file is 7. The access permission is set to 547
  - e. The UID of the file is Professor Pu-Jen Cheng. The GID of the file is 8. The access permission is set to 705
  
4. Which of the following is/are true about I/O performance?
  - a. A slow system call, such as one that reads data from the disk, may be blocked forever
  - b. A system call that waits for incoming network connections is a slow system call
  - c. A database system could use synchronous writes (`open()` with the `O_SYNC` flag) to make file updates faster
  - d. A system that enforces mandatory locking on file accesses may perform worse than a system that does not
  - e. The OS kernel could perform read ahead if sequential reads are detected

5. Which of the following is/are true about hard links and symbolic links?
- a. There is no actual file created in the file system when a new hard link is created
  - b. There is an actual file created for a symbolic link when a new symbolic link is created
  - c. One can use the command "ln" provided by your shell to create only symbolic links but not hard links
  - d. A symbolic link can link to a directory
  - e. A hard link can link to a file on a different file system
6. Which of the following is/are true about user identification?
- a. The OS kernel allocates a new UID whenever the user logs in
  - b. For any process, its real UID is always equal to the effective UID
  - c. The system searches for the default shell program to execute using the user's login ID (ex: root)
  - d. The process that executes a set-uid program will have its real UID updated
  - e. In Unix, different users must have unique UIDs. However, users may share the same GID
7. Which of the following is/are true about file I/Os?
- a. Consider two processes, A and B. Process A can pass its file descriptor to process B, so process B can use the file descriptor to access the file associated with the file descriptor
  - b. You can use the open() or creat() function to create a new file and specify the access permissions of the newly created file - the resulting file permissions of the file are always the same as specified by open() and creat()
  - c. Performing write() to a newly opened file with existing contents always writes the bytes from the beginning of the file (assuming the write() succeeds)
  - d. Assume a process opens a file a.txt successfully. It can then use lseek() alone to decrease the size of a.txt
  - e. Consider a case in which we read() 100 bytes from an opened regular file (size=50 bytes), with the current file offset set to 0. Assuming that the current read() returns 50. The next read from the file returns 0
8. Which of the following is/are true about the Unix file system?
- a. One can set the sticky bit of a directory to ensure any files within the directory are readable to only its respective owner
  - b. All open files of a given process share the same entry in the system open file table
  - c. The contents of a file are stored directly in the file's associated i-node
  - d. Assume that a program holds a file lock of a file a.txt but does not release the lock. After a process executes this program, no other program can ever open a.txt for reading
  - e. The device files are the files associated with hardware devices

**Short answer questions: (60 pts)**

- A. "Everything is a file" is a key feature in Unix. Explain why this simplifies programming efforts. (5 pts)
- B. Describe the key differences between blocking I/O, non-blocking I/O, and I/O multiplexing. Use an example to explain why using I/O multiplexing could provide better performance than the other two. (5 pts)



- C. Please refer to the figure above. Assume “foo” is a regular file located within the directory “/home/sp02/”. (12 pts in total, 3pts each)
- i1 has three i-node links. Explain why.
  - What does the figure look like after executing `unlink(foo)`?
  - What is the absolute pathname for “woo”?
  - Assume we want to create a symbolic “moo” in the directory where “woo” is within. Draw the changes in the figure after “moo” gets created.

- D. Alice implements a function, `setlock()`, that tries to acquire a write lock on the file and returns 0 on success. If any other processes have already held locks on the file, `setlock()` immediately returns -1. However, Bob comes and points out that the implementation below is buggy. Explain why. (6 pts)

```
int SP_FLOCK(int fd, int cmd, off_t start, off_t len) {
    struct flock fl;
    fl.l_type = F_WRLCK;
    fl.l_whence = SEEK_SET;
    fl.l_start = start;
    fl.l_len = len;
    if (cmd == F_SETLK)
        return fcntl(fd, cmd, &fl);
    else if (cmd == F_GETLK) {
        fcntl(fd, cmd, &fl);
        if (fl.l_type != F_UNLCK)
            fprintf(stderr, "lock held by pid %d\n", fl.l_pid);
        return fl.l_type;
    }
}

int setlock(int fd, off_t start, off_t len) {
    while (1) {
        // Assume that arguments fd, start, len are always valid
        ret = SP_FLOCK(fd, F_GETLK, start, len);
        if (ret == F_UNLCK) {
            ret = SP_FLOCK(fd, F_SETLK, start, len);
            if (!ret)
                return 0;
        } else
            return -1;
    }
}
```

- E. Answer the questions below. Assuming a process has executed the code below. (the path points to a regular file) (8 pts)

```
fd1 = open(path, oflags);  
fd2 = dup(f1);  
fd3 = open(path, oflags);  
link(path, path2); // path & path2 differ  
dup2(fd2, 1);
```

1. Which time values of the file specified by *path* is/are modified? (3 pts)
2. How many updates to the process' file descriptor table were made? (3 pts)
3. Assuming the same process calls `fcntl(fd1, F_SETFD, flags)` after `dup2()`, which descriptor(s) is/are affected? (2 pts)

- F. Why is running the following code dangerous? What does Unix do to limit the problem? (4 pts)

```
while (1) fork();
```

- G. Assume you are on the CSIE workstations. Your home directory is synchronized across all workstations, while other directories are not. The admins impose an upper limit on the number of processes that a single user is allowed to create. Suppose you have exceeded the limit on linux1.csie, and you're unable to login because the server refuses to create new processes for you. Meanwhile, other users are still able to run their programs successfully.

Given that you cannot get the admin privilege on linux1.csie or share your password with others. Describe the approach that you would take to re-enable login. (6pts)

- H. The code below is buggy. List at least two potential issues. (8 pts)

```
int main(int argc, char **argv) {
    if (argc < 2) exit(1);
    struct stat statbuf;
    char *buf;
    ssize_t count;
    int fd;
    fd = open(argv[1], O_RDONLY);
    if (fd < 0) exit(1);
    fstat(fd, &statbuf);
    buf = (char *)malloc(statbuf.st_size);
    while ((count = read(fd, buf, 1)) > 0)
        buf += count;
    return 0;
}
```

- I. Write a program in C that creates a zombie process. You should make sure that you can always find the zombie process via "ps" after the program is executed. You can ignore library headers. Make sure you use the parameters correctly. (6pts)