**Assignments**

**# I. General Topics**

BT1. Tạo static lib sử dụng Makefile

BT2. Tạo shared lib sử dụng Makefile

**# II. File system**

BT3. Tạo file test.txt và ghi một đoạn dữ liệu bất kì. Thực hiện ghi ra màn hình các thông tin: Loại file, tên file, thời gian chỉnh sửa file lần cuối, kích thước. (Sử dụng struct stat)

BT4. Viết một chương trình mở một file bất kì và sử dụng cờ O\_APPEND, sau đó thực hiện seek về đầu file rồi ghi một vài dữ liệu vào file đó. Dữ liệu sẽ xuất hiện ở vị trí nào của file và tại sao lại như vậy?

**# III. Process**

BT5. Viết một chương trình A tạo ra một tiến trình con B rồi in ra PID và PPID của chúng.

BT6. Viết một chương trình A tạo ra một tiến trình con B, in ra PID và PPID của chúng. Sử dụng system waitpid() để lấy được trạng thái kết thúc của tiến con B.

BT7. Từ kiến thức đã học, sử dụng SIGCHLD để ngăn ngừa việc hình thành zombie process.

**# IV. Signal**

BT8. Viết chương trình in ra thông điệp bất kì khi nhấn tổ hợp phím Ctrl+C. Đăng ký action cho SIGUSR1 và SIGUSR2.

BT9. Kiểm tra SIGINT có bị block trong process hay không. Nếu không thực hiện block/ublock thử SIGINT.

**# V. Multi-Thread**

BT10. Cung một ví dụ mà multi-thread không cung cấp hiệu suất tốt hơn các giải pháp single-thread

BT11. Viết một chương trình thực hiện tạo 2 threads. Cả hai threads này đều dùng chung một handler.

* Kiểm tra nếu là thread1 đang thực hiện thì in ra thông điệp bất kì để xác định.
* Nếu là thread2 thì truyền vào dữ liệu kiểu **struct human** được khởi tạo từ trước, với các thông tin: Họ tên, năm sinh, sdt, quê quán. Sau đó in các thông tin này ra màn hình.

BT12: Viết một chương trình thực hiện tạo 3 threads.

* Thread 1: Thực hiện việc nhập dữ liệu sinh viên từ bàn phím, bao gồm thông tin: Họ tên, ngày sinh, quê quán.
* Thread2: Mỗi lần nhập xong dữ liệu một sinh viên, thread này sẽ ghi thông tin sinh viên đó vào file (mỗi thông tin sinh viên nằm trên 1 dòng) thongtinsinhvien.txt.
* Thread3: Đọc dữ liệu vừa ghi được và in ra màn hình sau đó thông báo cho thread 1 tiếp tục nhập thêm sinh viên.

Sử dụng mutex và condition variable để giải quyết bài toán.

**#VI. Socket**

BT13: Viết chương trình client giao tiếp với server thông qua socket sử dụng ipv4 stream socket.

**#VII. Pipe and FIFO (Named Pipe)**

BT14: Tạo Pipe, trao đổi dữ liệu giữa các related process.

BT15: Tạo FIFO, trao đổi dữ liệu giữa các unrelated process.

**#VII. Shared Memory**

BT16: So sánh giữa hai phương thức mapping memory sử dụng hàm mmap() (POSIX api). Triển khai code để chứng minh.

#VIII. Message Queue  
BT17. Tạo ra một danh sách sinh viên mới message queues. Định dạng dữ liệu lưu trữ trong queues như sau:

struct message {

long priority <id của sinh viên>;

struct sinhvien {

char hoten[50];

int tuoi;

char quequan[100];

char sdt[100];

};

}

* Menu hiển thị chức năng.
* Nhập thông tin sinh viên từ bàn phím và lưu trữ vào trong queue.
* Hiển thị thông tin sinh viên thông qua id.
* Hiển thị toàn bộ danh sách của sinh viên.
* Xóa một sinh viên khỏi danh sách thông qua id.
* Lưu ý: giá trị id của mỗi sinh viên là duy nhất và lớn hơn 0.

**#VIII**. Tạo một character device driver (kernel space) và một chương trình trên user space thực hiện việc ghi và đọc dữ liệu vào device file tương ứng với character device driver đó. Lưu ý, có thể viết và chạy trực tiếp trên Ubuntu.