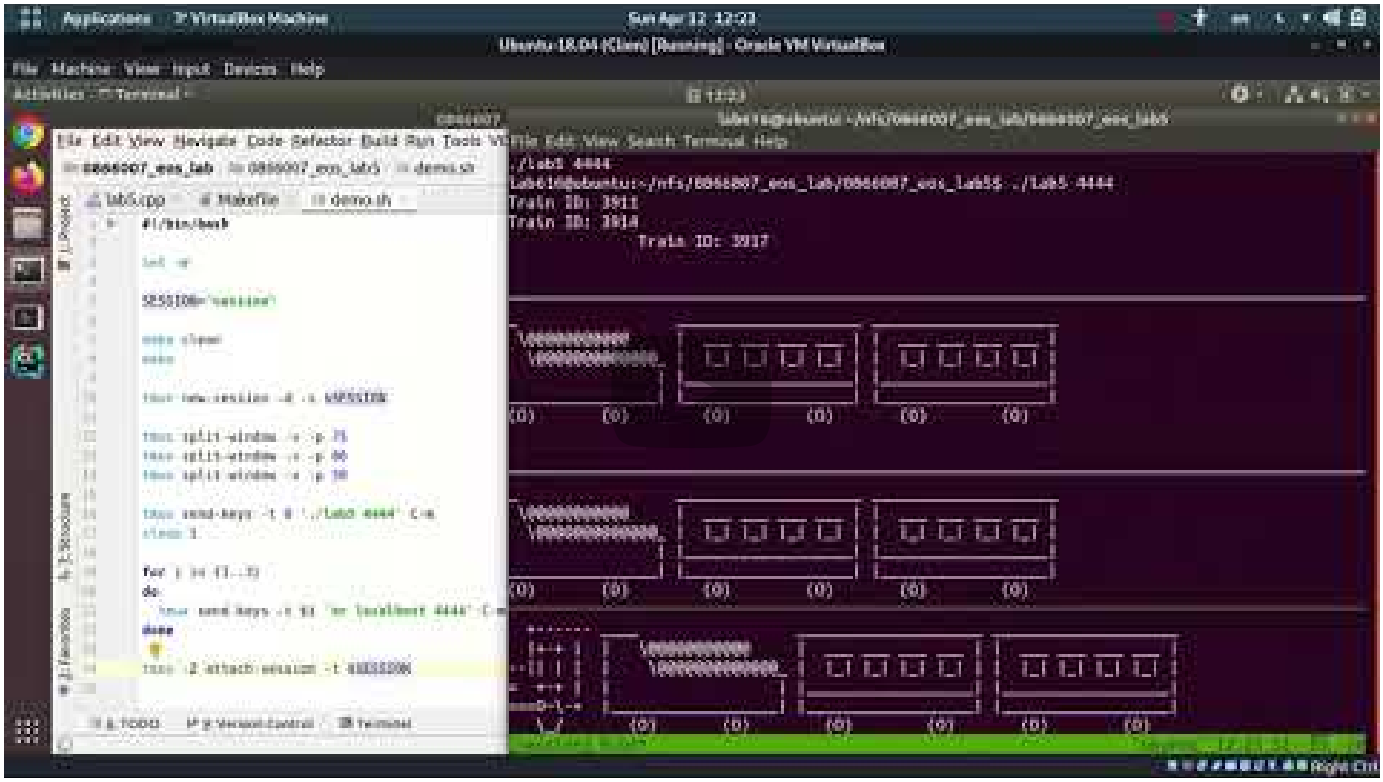
Lab 5 東方快車



I. Introduction

請在 VM 上撰寫一隻Socket程式。在 VM 上會先將 server 程式執行起來，等待 client 程式來做連線。當連線建立之後，server 會將列車的 ASCII art，透過 socket 傳給 client 端，於是在 client 端 terminal 上就會看見一台列車從右到左急駛而過。

II. Specification

- Server (**自行撰寫**)
  - 建立 socket server，等待 client 來建立連線。
  - server 端要能夠支援同時多人連線的情境。
  - 連線建立後，會去執行 **sl** 程式並將輸出透過 socket 傳給 client 端。
  - 在 server 上印出火車的 ID，用建立出的 process id 來表示。
- Client (助教提供)
  - 使用 netcat 這隻程式來作為 client 端。
  - 與 server 建立連線後，接收傳來的資料。

III. Illustration

- Server

```
./lab5 <port>
```

- <port> 為 server listen 的端口。

- Client

```
nc <ip> <port>
```

- <ip> 為 server 所在的 ip。
- <port> 為 server listen 的端口。

IV. Note

- socket 與 process 的撰寫請參考 laball.pdf 之 lab 3， 6-2 的範例程式碼。
- VM 上須安裝 sl, tmux 程式，以利程式正確執行。

```
sudo apt install sl tmux
```

- 先ctrl-b，再用指令來關閉已經打開的session(影片最後的步驟)

```
:kill-session
```

- 在執行 sl 程式時，請加上參數 `-l` 以產生出較小的列車。

```
sl -l
```

- 在 server 程式碼當中
  - 請使用 execlp() (https://linux.die.net/man/3/execlp) 來執行 sl 程式。
  - 請使用 dup2() (http://man7.org/linux/man-pages/man2/dup.2.html) system call 將標準輸出重新導向給 socket 的 fd。
- 請將下列程式碼加入到你的程式碼當中，來清除執行過程中產生的 zombie process。

```
#include <signal.h>
#include <sys/wait.h>

void handler(int signum) {
    while (waitpid(-1, NULL, WNOHANG) > 0);
}

int main(int argc, char *argv[]) {
    signal(SIGCHLD, handler);
}
```

V. Demo & Submission

- 助教會提供 demo.sh (http://demo.sh) 來協助同學測試與 Demo。
- 助教會透過下列指令檢查同學是否有在系統上留下 zombie process。

```
ps aux | grep defunct | grep -v grep
```

- 請將程式碼以下列的格式擺放與命名，以方便助教評分。

```
<學號>_eos_lab5
|-- Makefile
|-- lab5.cpp
|-- demo.sh
```

- 請將上述之資料夾壓縮為單一 zip 檔案，並上傳到 E3 上。

Lab 5 Hints

I. Preliminaries

- Review Lab 6-2: Socket Programming and Lab 8: Signal and Timer in laball

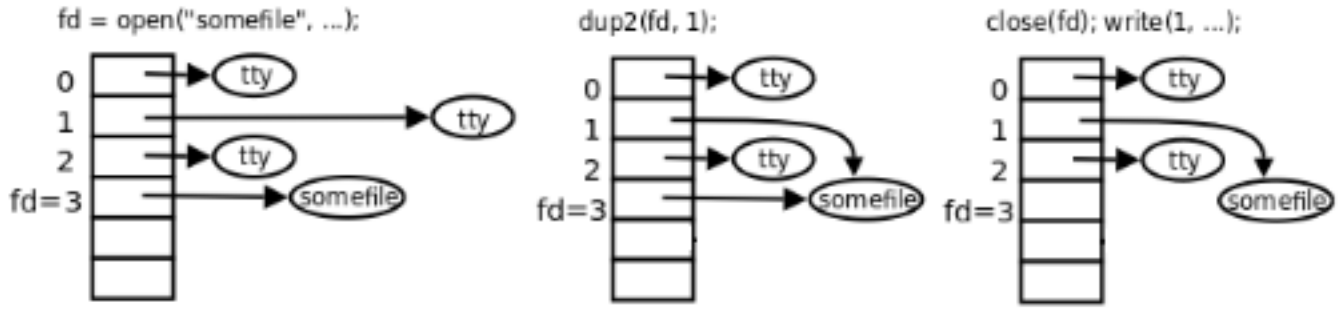
II. dup2 - duplicate a file descriptor

```
#include <stdio.h>  // perror()
#include <stdlib.h> // exit()
#include <fcntl.h>  // open()
#include <unistd.h> // dup2()

int main(int argc, char *argv[]) {
        int fd;
        if ((fd = open("test.txt", O_CREAT | O_RDWR, 0644)) == -1) {
                perror("open");
                exit(EXIT_FAILURE);
        }

        dup2(fd, STDOUT_FILENO);
        close (fd);

        printf("Hello, World!\n");
        return 0;
}
```



(https://asciinema.org/a/2zGPw2BU35azHObMAGAfydBSm)

III. sl - display animations

```
sudo apt install sl
```

(https://asciinema.org/a/aOaPI1upcloMqIfhFu7RHdEKC)

IV. exec - Execute a file

- execl, execlp, execv, execvp
  - int execl(const char *path, const char *arg, ...);
  - int execlp(const char *file, const char *arg, ...);
  - int execv(const char *path, char *const argv[]);
  - int execvp(const char *file, char *const argv[]);
- Example for execute ls instruction

```
#include <unistd.h>

int main(int argc, char *argv[]) {
    execl("/bin/ls", "/bin/ls", "-l", NULL);

    execlp("ls", "ls", "-l", NULL);

    char *arg1[] = {"/bin/ls", "-l", NULL};
    execv(arg1[0], arg1);

    char *arg2[] = {"ls", "-l", NULL};
    execvp(arg2[0], arg2);
}
```

V. Zombie process handler

- Example for waiting zombie process

```
#include <signal.h>   // signal()
#include <sys/wait.h> // waitpid()

void handler(int signum) {
    while (waitpid(-1, NULL, WNOHANG) > 0);
}

int main(int argc, char *argv[]) {
    signal(SIGCHLD, handler);
}
```

VI. Address already in use

- Force using socket address already in use

```
#include <sys/socket.h>

int main(int argc, char *argv[]) {
    int fd = socket(...);

    int yes = 1;
    setsockopt(fd, SOL_SOCKET, SO_REUSEADDR, &yes, sizeof(yes));

    bind(fd, ...)
}
```

VII. Close server socket

- Close socket when catching SIGINT signal

```
#include <signal.h> // signal()
#include <unistd.h> // close()

int sockfd;

void handler(int signum) {
    close(sockfd);
}

int main(int argc, char *argv[]) {
    signal(SIGINT, handler);
}
```