網路程式設計 作業一　學號/系級/姓名　4054100990/資工三/陳知余

一、fork_server.c:

(1)程式流程圖：

1.設定好 server 的 address type：

```c
int status;
struct addrinfo hints;
struct addrinfo *servinfo, *ptr; // point to the result and a addrinfo pointer
memset(&hints, 0, sizeof(hints)); // ensure the struct is empty
hints.ai_family = AF_UNSPEC; // IPv4 or IPv6
hints.ai_socktype = SOCK_STREAM; // TCP stream sockets
hints.ai_flags = AI_PASSIVE; // fill up my IP
if ((status = getaddrinfo(NULL, PORT, &hints, &servinfo)) != 0) {
  fprintf(stderr, "getaddrinfo error: %s\n", gai_strerror(status));
  exit(1);
```

2.接著跑一個 for 迴圈，分別是 socket 函式設定 socket，setsockopt 設定 socket option，bind 函式定義在預設的 port(設定 80)，如果都沒 error 就 break。

3. 在設定好的 fd listen，最大限制連線 RESTRICT 設定為 10。

```c
if (listen(server_fd, RESTRICT) == -1) {  //listen at port 80, the limit is RESTRICT(a constant)
  perror("listen");
  exit(1);
}
```

4.設定 signal handler，避免 zombie 產生(原理如圖中註解)。

```c
/*deal with the zombie process using the waitpid(),
then the parent process have to wait until the child process end.
Thus, the kernel can get the info and release the pid*/
struct sigaction sa;
sa.sa_handler = sigchld_handler;
sigemptyset(&sa.sa_mask);
sa.sa_flags = SA_RESTART;

if (sigaction(SIGCHLD, &sa, NULL) == -1) {
  perror("sigaction");
  exit(1);
}
```

```c
void sigchld_handler(int s)
{
  while (waitpid(-1, NULL, WNOHANG) > 0)
    ;
}
```

5.最後進入主要的 while loop，透過 fork()產生 child process 提供 client 連線 server(>0:child pid, ==0:child, <0:error)，並用 read 把 client_fd 的內容讀入再輸出，網頁部分原本是想用 fopen 讀 html 檔再傳送，但發現會有問題(網路上也說不要用這種方法)，所以直接在 code 裡令一個 char array 取代，最後也順利顯示。

```c
  int pid;
  while(1) {
    client_fd = accept(server_fd, (struct sockaddr *)&client_addr, &addr_len);

    if(client_fd == -1) {
      perror("Connection failed\n");
      continue;
    }
    printf("Got client connection from %s\n", inet_ntoa(client_addr.sin_addr));
    pid = fork();
    if (pid == 0) {
      /*child process */
      close(server_fd);
      memset(buf, '\0', sizeof(buf));
      read(client_fd, buf, sizeof(buf) - 1);

      printf("%s\n", buf);

      write(client_fd, website, sizeof(website) -1);
      close (client_fd);
      printf("Closing the client_fd\n");
      exit(0);
    }
    /*parent process*/
    if (pid > 0) {
      wait(NULL);
      close(client_fd);
    }
  }
  return 0;
}
```

```c
char website[] =
"HTTP/1.1 200 OK\r\n"
"Content-Type: text/html; charset=UTF-8\r\n\r\n"
"<!DOCTYPE html>\r\n"
"<html>\r\n"
"<head>\r\n"
"<meta name=\"viewport\" content=\"width=device-width, minimum-scale=0.1\">\r\n"
"<title>0Dw4dCP.png (1200×1678)</title></head><body style=\"margin: 0px; background: #0e0e0e;\">\r\n"
"<img style=\"-webkit-user-select: none;cursor: zoom-in;\" src=\"https://i.imgur.com/0Dw4dCP.png\" width=\"379\" height=\"530\">\r\n"
"</body></html>\r\n"; /*html website*/
```

(2)功能實作技巧

Socket, setsockopt, bind, listen 要照順序使用，用一個 for 迴圈才能保證使用的 fd 在這些函式沒有出現 error。

(3)重要資料結構

1.struct addrinfo {

int ai_flags;

int ai_family;

int ai_socktype;

int ai_protocol;

socklen_t ai_addrlen;

struct sockaddr *ai_addr;

char *ai_canonname;

struct addrinfo *ai_next;

};

作業中有使用的是

ai_family:設定 IPv4 或 IPv6(AF_INET and AF_INET6)，不確定就用 AF_UNSPEC 等 getaddrinfo()回傳資訊。。

ai_socktype:設定 socket 的 type，例如 SOCK_STREAM (TCP Socket) or SOCK_DGRAM(UDP Socket)。

ai_flags:設定額外特殊的選項

(4)期望加分特點或新功能

1.利用 read 把連線的 client 資訊讀入 buffer 再印出。

2.透過 inet_ntoa 顯示連線者的 address。

3.使用 signal handler 讓 parent process wait 直到 child process 結束，才能處理 child process 被結束後變成 zombie 的可能。

二、select_server.c:

(1)程式流程圖：

1.基本上前面設定 server 的過程和 fork_server.c 一樣

```c
struct addrinfo hints;
struct addrinfo *servinfo, *ptr; // point to the result and a addrinfo pointer
memset(&hints, 0, sizeof(hints)); // ensure the struct is empty
hints.ai_family = AF_UNSPEC; // IPv4 or IPv6
hints.ai_socktype = SOCK_STREAM; // TCP stream sockets
hints.ai_flags = AI_PASSIVE; // fill up my IP
/* Translate name of a service location and/or a service name to set of
 socket addresses.

 This function is a possible cancellation point and therefore not
 marked with __THROW.  */
if ((status = getaddrinfo(NULL, PORT, &hints, &servinfo)) != 0) {
  fprintf(stderr, "getaddrinfo error: %s\n", gai_strerror(status));
  exit(1);
}
int listen_fd, optval = 1;
```

```c
int listen_fd, optval = 1;
for (ptr = servinfo; ptr != NULL; ptr = ptr->ai_next) {
  /* Create a new socket of type TYPE in domain DOMAIN, using
  protocol PROTOCOL.  If PROTOCOL is zero, one is chosen automatically.
  Returns a file descriptor for the new socket, or -1 for errors.  */
  if ((listen_fd = socket(ptr->ai_family, ptr->ai_socktype, ptr->ai_protocol)) < 0) { //set socket
    perror("socket error:");
    continue;
  }
  /* Set socket FD's option OPTNAME at protocol level LEVEL
  to *OPTVAL (which is OPTLEN bytes long).
  Returns 0 on success, -1 for errors.  */
  if (setsockopt(listen_fd, SOL_SOCKET, SO_REUSEADDR, &optval, sizeof(int)) < 0) { //set socket option
    perror("set socket option error:");
    exit(1);
  }
  /* Give the socket FD the local address ADDR (which is LEN bytes long).  */
  if (bind(listen_fd, ptr->ai_addr, ptr->ai_addrlen) < 0) {
    perror("bind error:");
    close(listen_fd);
    continue;
  }
  break;
}
if (ptr == NULL) {  //failed to bind
  fprintf(stderr, "elect_server: failed to bind\n");
  exit(2);
}
freeaddrinfo(servinfo);
/* Prepare to accept connections on socket FD.
 N connection requests will be queued before further requests are refused.
 Returns 0 on success, -1 for errors.  */
if (listen(listen_fd, RESTRICT) == -1) {
  perror("listen error");
  exit(3);
}
```

比較不同的是多了 fd_set 的 data type，能透過 FD_SET 將合格的 socket descriptors 新增到 set 中，讓之後的 select 來選擇合格的連線。

(FD_ZERO:清零)

```c
fd_set main_fds, read_fds;  //main fd list && temporary fd list
FD_ZERO(&main_fds); FD_ZERO(&read_fds);
```

(FD_SET:將 fd 加到 set)

```c
FD_SET(listen_fd, &main_fds);
int max_fd = listen_fd;
```

2.接著一樣進入 while 迴圈，首先令一個 struct timeval，用來設定 select 執行間隔的時間。

int select(int numfds, fd_set *readfds, fd_set *writefds,

fd_set *exceptfds, struct timeval *timeout);

readfds、writefds 及 exceptfds 分別表示要監視的那一組 file descriptor set。

如果成功，select() 回傳 number of file descriptors contained in the three returned descriptor sets，如果等於 0 則代表時間到期在事件發生之前，-1 則是有 error。

剩下如果 fd 是新的連線，就用和 fork 一樣的 accept 將網頁傳給 client，如果是已連線的狀況則改用 excv()傳送，最後用 FD_CLR()將該 fd 移出 set。

```c
while (1) {
  fd_set read_fds = main_fds;
  struct timeval tv;
  tv.tv_sec = 3; tv.tv_usec = 141592;
  int ret = 0, i = 0;
  /* Check the first NFDS descriptors each in READFDS (if not NULL) for read
 readiness, in WRITEFDS (if not NULL) for write readiness, and in EXCEPTFDS
 (if not NULL) for exceptional conditions.  If TIMEOUT is not NULL, time out
 after waiting the interval specified therein.  Returns the number of ready
 descriptors, or -1 for errors.

 This function is a cancellation point and therefore not marked with
 __THROW.  */
  if ((ret = select(max_fd + 1, &read_fds, NULL, NULL, &tv)) == -1) {
    perror("select error:");
    exit(4);
  } else if (ret == 0) {
    printf("Time out!\n");
    continue;
  } else {
    for (i = 0; i < max_fd; i++) {
      if (FD_ISSET(i, &read_fds)) {
        if (i == listen_fd) {  //new connected
          struct sockaddr_in client_addr;
          int new_fd;
          socklen_t socklen = sizeof(struct sockaddr_in);
          /* Await a connection on socket FD.
          When a connection arrives, open a new socket to communicate with it,
          set *ADDR (which is *ADDR_LEN bytes long) to the address of the connecting
          peer and *ADDR_LEN to the address's actual length, and return the
          new socket's descriptor, or -1 for errors.

          This function is a cancellation point and therefore not marked with
          __THROW.  */
          if ((new_fd = accept(listen_fd, (struct sockaddr *)&client_addr, &socklen) == -1)) {
            perror("accept error:");
            exit(5);
```

```c
          new socket's descriptor, or -1 for errors.

          This function is a cancellation point and therefore not marked with
          __THROW.  */
          if ((new_fd = accept(listen_fd, (struct sockaddr *)&client_addr, &socklen) == -1)) {
            perror("accept error:");
            exit(5);
          } else {
            memset(buf, '\0', sizeof(buf));
            read(new_fd, buf, sizeof(buf) - 1);
            printf("New connection comes from %s on PORT:%s by fd:%d\n",
                   inet_ntoa(client_addr.sin_addr), PORT, new_fd);
            write(new_fd, website, sizeof(website) - 1);
          }
        } else { // already connected
          int recv_len;
          memset(buf, '\0', sizeof(buf));
          //recv() is used to receive messages from a connected socket//
          if ((recv_len = recv(i, buf, sizeof(buf), 0)) == -1) {
            perror("recv error:");
            exit(6);
          } else if (recv_len == 0) {
            printf("Disconnected\n");
          } else {
            printf("Receive message:\n%s\n", buf);
            send(i, buf, recv_len, 0);
          }
          close(i);
          FD_CLR(i, &main_fds);
        }
      }
    }
  }
}
return 0;
}
```

(2)功能實作技巧:

和 fork_server.c 一樣，server 設定時 Socket, setsockopt, bind, listen 要照順序使

用，另外 select 的最後一個參數可以設定時間間隔，如果設定為 0，select() 會在輪詢問過 sets 中的每個 file descriptors 之後就 timeout。如果設定為 NULL，就永遠不會 timeout 並且陷入等待，直到至少一個 file descriptor 已經準備好了。

(3)重要資料結構

```
1. struct timeval {
      time_t      tv_sec;     /* seconds */
      suseconds_t tv_usec;    /* microseconds */
};
```

定義在<time.h>裡，前者單位是秒，後者單位是微秒，

1 microsecond = 1 /1000000 second.

```
2. struct sockaddr_in {
                sa_family_t    sin_family; /* address family: AF_INET */
                in_port_t      sin_port;   /* port in network byte order */
                struct in_addr sin_addr;   /* internet address */
            };
```

主要是給 IPv4 用的。

(4)期望加分特點或新功能：

1.有對幾乎全部的函數做 error 檢查，並使用 perror 印出錯誤訊息。

2.有查詢的函數都有附上註解方便 review 功能、參數、return value…等等。