

COMP2303/7306 Week 12 Code Examples

To be commented in class.

Sample select() Server

```
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <stdio.h>
#include <stdlib.h>      /* For atoi() */
#include <ctype.h>      /* For toupper() */
#include <unistd.h>     /* For read() */

int open_listen(int port)
{
    int fd;
    struct sockaddr_in serverAddr;
    int optVal;

    fd = socket(AF_INET, SOCK_STREAM, 0);
    if(fd < 0) {
        perror("Error creating socket");
        exit(1);
    }

    optVal = 1;
    if(setsockopt(fd, SOL_SOCKET, SO_REUSEADDR, &optVal, sizeof(int)) < 0) {
        perror("Error setting socket option");
        exit(1);
    }

    serverAddr.sin_family = AF_INET;
    serverAddr.sin_port = htons(port);
    serverAddr.sin_addr.s_addr = htonl(INADDR_ANY);

    if(bind(fd, (struct sockaddr*)&serverAddr, sizeof(struct sockaddr_in)) < 0){
        perror("Error binding socket to port");
        exit(1);
    }
    if(listen(fd, SOMAXCONN) < 0) {
        perror("Error listening");
        exit(1);
    }

    return fd;
}

char* capitalise(char* buffer, int len)
{
    int i;

    for(i=0; i<len; i++) {
        buffer[i] = (char)toupper((int)buffer[i]);
    }
    return buffer;
}
```

```

void process_connections(int fdServer)
{
    int fd;
    int maxFD;
    int newFD;
    struct sockaddr_in fromAddr;
    int fromAddrSize;
    char buffer[1024];
    ssize_t numBytesRead;
    fd_set baseSet, readSet;

    maxFD = fdServer;

    FD_ZERO(&baseSet);
    FD_SET(fdServer, &baseSet);
    while(1) {
        readSet = baseSet;

        if(select(maxFD + 1, &readSet, NULL, NULL, NULL) < 0) {
            perror("Select failed");
            exit(1);
        }
        for(fd = maxFD; fd >= 0; fd--) {
            if(FD_ISSET(fd, &readSet)) {
                if(fd == fdServer) {
                    fromAddrSize = sizeof(struct sockaddr_in);
                    newFD = accept(fdServer, (struct sockaddr*)&fromAddr,
                                   &fromAddrSize);
                    if(newFD < 0) {
                        perror("Error accepting connection");
                        exit(1);
                    }
                    FD_SET(newFD, &baseSet);
                    if(newFD > maxFD) {
                        maxFD = newFD;
                    }
                    printf("Accepted request from %s, port %d\n",
                           inet_ntoa(fromAddr.sin_addr),
                           ntohs(fromAddr.sin_port));
                } else {
                    numBytesRead = read(fd, buffer, 1024);
                    if(numBytesRead > 0) {
                        fflush(stdout);
                        capitalise(buffer, numBytesRead);
                        write(fd, buffer, numBytesRead);
                    } else if(numBytesRead < 0) {
                        perror("Error reading from socket");
                        exit(1);
                    } else {
                        FD_CLR(fd, &baseSet);
                        if(fd == maxFD) {
                            do {
                                maxFD--;
                            } while (!FD_ISSET(maxFD, &baseSet));
                        }
                        close(fd);
                    }
                }
            }
        }
    }
}

```

```
int main(int argc, char* argv[])
{
    int portnum;
    int fdServer;

    if(argc != 2) {
        fprintf(stderr, "Usage: %s port-num\n", argv[0]);
        exit(1);
    }

    portnum = atoi(argv[1]);
    if(portnum < 1024 || portnum > 65535) {
        fprintf(stderr, "Invalid port number: %s\n", argv[1]);
        exit(1);
    }

    fdServer = open_listen(portnum);

    process_connections(fdServer);

    return 0;
}
```

chat.c

```
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

int create_and_bind_socket(port)
{
    int fd;
    struct sockaddr_in myAddress;

    if((fd = socket(AF_INET, SOCK_DGRAM, 0)) < 0) {
        perror("Error creating socket");
        exit(1);
    }

    myAddress.sin_family = AF_INET;
    myAddress.sin_port = htons(port);
    myAddress.sin_addr.s_addr = INADDR_ANY;

    if(bind(fd, (struct sockaddr *)&myAddress, sizeof(struct sockaddr_in)) < 0){
        perror("Error binding socket to port");
        exit(1);
    }
    return fd;
}

void communicate(fd, destPort)
{
    fd_set readSet;
    int numBytes;
    char buffer[1024];
    struct sockaddr_in theirAddress, destAddress;
    int theirAddrSize;

    destAddress.sin_family = AF_INET;
    destAddress.sin_port = htons(destPort);
    destAddress.sin_addr.s_addr = inet_addr("127.0.0.1");

    FD_ZERO(&readSet);
```

```

while(1) {
    FD_SET(fd, &readSet);
    FD_SET(STDIN_FILENO, &readSet);

    if(select(fd + 1, &readSet, NULL, NULL, NULL) < 0) {
        perror("Select failed");
        exit(1);
    }

    if(FD_ISSET(STDIN_FILENO, &readSet)) {
        numBytes = read(STDIN_FILENO, buffer, 1024);
        if(numBytes > 0) {
            if(sendto(fd, buffer, numBytes, 0,
                (struct sockaddr *)&destAddress,
                sizeof(struct sockaddr_in)) < 0) {
                perror("sendto failed");
                exit(1);
            }
        } else if(numBytes == 0) {
            close(fd);
            exit(1);
        } else {
            perror("Error reading from standard input");
            exit(1);
        }
    }
    if(FD_ISSET(fd, &readSet)) {
        theirAddrSize = sizeof(struct sockaddr_in);
        numBytes = recvfrom(fd, buffer, 1024, 0,
            (struct sockaddr *)&theirAddress,
            &theirAddrSize);
        if(numBytes > 0) {
            printf("From %s:%d: ", inet_ntoa(theirAddress.sin_addr),
                ntohs(theirAddress.sin_port));
            fflush(stdout);
            write(STDOUT_FILENO, buffer, numBytes);
        } else if(numBytes == 0) {
            close(fd);
            exit(1);
        } else {
            perror("Error in recvfrom");
            exit(1);
        }
    }
}
}

int main(int argc, char* argv[])
{
    int myPort, destPort, fd;

    if(argc != 3) {
        printf("Usage: %s my-port dest-port\n", argv[0]);
        exit(1);
    }
    myPort = atoi(argv[1]);
    destPort = atoi(argv[2]);
    fd = create_and_bind_socket(myPort);
    communicate(fd, destPort);
    return 0;
}

```