

## COMP2303/7306 Week 9 Friday Code Examples

To be commented in class.

### Sample Multi-process Server

```
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include <unistd.h>
#include <netdb.h>
#include <sys/wait.h>
#include <signal.h>
#include <errno.h>

#define MAXHOSTNAMELEN 128

void child_sig_handler(int sig)
{
    pid_t pid;

    while((pid = waitpid(-1, NULL, WNOHANG)) > 0) {
        printf("Child done (%d)\n", pid);
        fflush(stdout);
    }
}

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, error;
    struct sockaddr_in fromAddr;
    socklen_t fromAddrSize;
    char hostname[MAXHOSTNAMELEN];
    char buffer[1024];
    ssize_t numBytesRead;
    pid_t pid;

    while(1) {
        fromAddrSize = sizeof(struct sockaddr_in);

        while((fd = accept(fdServer, (struct sockaddr*)&fromAddr,
            &fromAddrSize)) < 0) {
            if(errno != EINTR) {
                perror("Error accepting connection");
                exit(1);
            }
        }

        error = getnameinfo((struct sockaddr*)&fromAddr, fromAddrSize, hostname,
            MAXHOSTNAMELEN, NULL, 0, 0);
        if(error) {
            fprintf(stderr, "Error getting hostname: %s\n", gai_strerror(error));
        } else {
            printf("Accepted connection from %s (%s), port %d\n",
                inet_ntoa(fromAddr.sin_addr),
                hostname, ntohs(fromAddr.sin_port));
        }
        fflush(stdout);

        pid = fork();
        if (pid == 0) {
            close(fdServer);

            while((numBytesRead = read(fd, buffer, 1024)) > 0) {
                capitalise(buffer, numBytesRead);
                write(fd, buffer, numBytesRead);
            }

            if(numBytesRead < 0) {
                perror("Error reading from socket");
                exit(1);
            }
            printf("Done\n");
            fflush(stdout);
            close(fd);
            exit(0);
        } else if(pid > 0) {
            close(fd);
        } else {
            perror("Error forking");
            exit(1);
        }
    }
}

```

```

int main(int argc, char* argv[])
{
    int portnum;
    int fdServer;
    struct sigaction sa;

    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);
    }

    sa.sa_handler = child_sig_handler;
    sa.sa_flags = SA_RESTART | SA_NOCLDSTOP;
    sigaction(SIGCHLD, &sa, NULL);
    fdServer = open_listen(portnum);
    process_connections(fdServer);
    return 0;
}

```

## Sample Multi-threaded Server

```

#include <...> /* Much the same as above */
#include <pthread.h>

#define MAXHOSTNAMELEN 128

void* client_thread(void* arg);

int open_listen(int port)
{
    ... /* Same as above */
}

char* capitalise(char* buffer, int len)
{
    ... /* Same as above */
}

void process_connections(int fdServer)
{
    int fd;
    struct sockaddr_in fromAddr;
    socklen_t fromAddrSize;
    int error;
    char hostname[MAXHOSTNAMELEN];
    pthread_t threadId;

    while(1) {
        fromAddrSize = sizeof(struct sockaddr_in);

        fd = accept(fdServer, (struct sockaddr*)&fromAddr, &fromAddrSize);
        if(fd < 0) {
            perror("Error accepting connection");
            exit(1);
        }
        error = getnameinfo((struct sockaddr*)&fromAddr, fromAddrSize, hostname,
            MAXHOSTNAMELEN, NULL, 0, 0);
    }
}

```

```

        if(error) {
            fprintf(stderr, "Error getting hostname: %s\n",
                    gai_strerror(error));
        } else {
            printf("Accepted connection from %s (%s), port %d\n",
                    inet_ntoa(fromAddr.sin_addr),
                    hostname,
                    ntohs(fromAddr.sin_port));
        }

        pthread_create(&threadId, NULL, client_thread, (void*)fd);
        pthread_detach(threadId);
    }
}

void* client_thread(void* arg)
{
    int fd;
    char buffer[1024];
    ssize_t numBytesRead;

    fd = (int)arg;

    while((numBytesRead = read(fd, buffer, 1024)) > 0) {
        capitalise(buffer, numBytesRead);
        write(fd, buffer, numBytesRead);
    }

    if(numBytesRead < 0) {
        perror("Error reading from socket");
        exit(1);
    }
    printf("Done\n");
    fflush(stdout);
    close(fd);
    pthread_exit(NULL);
    return NULL;
}

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;
}

```