ID: 315144972_207988213

מטלה 2 מערכות הפעלה

תרגיל 1

```
mastik@mastik-VirtualBox:~/ComputerScience/OperatingSystems/Ex2/prog1$ cat ttt.cpp.gcov
        -:    0:Source:ttt.cpp
        -:    0:Graph:ttt.gcno
        -:    0:Data:ttt.gcda
        -:    0:Runs:12
        -:    1:#include <iostream>
        -:    2:#include <algorithm>
        -:    3:#include <cctype>
        -:    4:#include "ttt.hpp"
        -:    5:
        -:    6:using namespace std;
        -:    7:
       10:    8:TicTacToe::TicTacToe(const string& strategy) {
       10:    9:if (!validateStrategy(strategy)) {
        5:   10:    cout << "Error\n";
        5:   11:    exit(1);
        -:   12:}
        -:   13:this->strategy = strategy;
        5:   14:board = vector<char>(9, ' ');
        5:   15:}
        -:   16:
        5:   17:void TicTacToe::playGame() {
        -:   18:while (true) {
       17:   19:    int move = makeMove();
       17:   20:    board[move - 1] = 'X';
       17:   21:    cout << move << "\n";
       17:   22:    if (checkWin('X')) {
        2:   23:        cout << "I Won\n";
        4:   24:        break;
        -:   25:    }
       15:   26:    if (isBoardFull()) {
        1:   27:        cout << "DRAW\n";
        -:   28:        break;
        -:   29:    }
        -:   30:
        -:   31:    int playerMove;
       14:   32:    cin >> playerMove;
       14:   33:    if (!isValidMove(playerMove)) {
        1:   34:        cout << "Error\n";
        1:   35:        exit(1);
        -:   36:    }
```

```
    -:   36:    }
   13:   37:    board[playerMove - 1] = 'O';
   13:   38:    if (checkWin('O')) {
    1:   39:        cout << "I Lost\n";
    -:   40:        break;
    -:   41:    }
   12:   42:    if (isBoardFull()) {
#####:   43:        cout << "DRAW\n";
    -:   44:        break;
    -:   45:    }
   12:   46:}
    4:   47:}
    -:   48:
   10:   49:bool TicTacToe::validateStrategy(const string& strategy) {
   10:   50:if (strategy.size() != 9) return false;
    7:   51:vector<int> count(10, 0);
   65:   52:for (char c : strategy) {
   59:   53:    if (!isdigit(c) || c == '0') return false;
   58:   54:    count[c - '0']++;
    -:   55:}
    6:   56:return all_of(count.begin() + 1, count.end(), [](int i) { return i == 1; });
    -:   57:}
    -:   58:
#####:   59:bool TicTacToe::isValidMove(int move) {
  14*:   60:return move >= 1 && move <= 9 && board[move - 1] == ' ';
    -:   61:}
    -:   62:
   17:   63:int TicTacToe::makeMove() {
   56:   64:for (char c : strategy) {
   56:   65:    int move = c - '0';
   56:   66:    if (board[move - 1] == ' ') {
    -:   67:        return move;
    -:   68:    }
    -:   69:}
    -:   70:return -1;
    -:   71:}
    -:   72:
   30:   73:bool TicTacToe::checkWin(char player) {
    -:   74:const vector<vector<int>> winPositions = {
    -:   75:    {0, 1, 2}, {3, 4, 5}, {6, 7, 8}, // rows
    -:   76:    {0, 3, 6}, {1, 4, 7}, {2, 5, 8}, // columns
    -:   77:    {0, 4, 8}, {2, 4, 6}             // diagonals
  300:   78:};
  257:   79:for (const auto& pos : winPositions) {
  230:   80:    if (board[pos[0]] == player && board[pos[1]] == player && board[pos[2]] == player) {
    -:   81:        return true;
    -:   82:    }
    -:   83:}
    -:   84:return false;
   30:   85:}
    -:   86:
   27:   87:bool TicTacToe::isBoardFull() {
   27:   88:return all_of(board.begin(), board.end(), [](char c) { return c != ' '; });
    -:   89:}
    -:   90:
    -:   91:
   12:   92:int main(int argc, char* argv[]) {
   12:   93:if (argc != 2) {
    2:   94:    std::cout << "Error\n";
    2:   95:    return 1;
    -:   96:}
   10:   97:std::string strategy = argv[1];
   10:   98:TicTacToe game(strategy);
    5:   99:game.playGame();
    -:  100:return 0;
    4:  101:}
tik@mastik-VirtualBox:~/ComputerScience/OperatingSystems/Ex2/prog1$
```

תרגיל 2

```
mastik@mastik-VirtualBox:~/ComputerScience/OperatingSystems/Ex2/prog2$ cat mync.cpp.gcov
        -:    0:Source:mync.cpp
        -:    0:Graph:mync.gcno
        -:    0:Data:mync.gcda
        -:    0:Runs:4
        -:    1:#include <iostream>
        -:    2:#include <string>
        -:    3:#include <vector>
        -:    4:#include <unistd.h>
        -:    5:#include <sys/types.h>
        -:    6:#include <sys/wait.h>
        -:    7:#include <sstream>
        -:    8:
        3:    9:void executeProgram(const std::string& command) {
        -:   10:// Split the command into program and arguments
        3:   11:std::istringstream iss(command);
        3:   12:std::string token;
        3:   13:std::vector<std::string> tokens;
        -:   14:
        9:   15:while (iss >> token) {
        6:   16:    tokens.push_back(token);
        -:   17:}
        -:   18:
        3:   19:std::vector<char*> args;
        9:   20:for (const std::string& s : tokens) {
        6:   21:    char* arg = new char[s.size() + 1];
        6:   22:    std::copy(s.begin(), s.end(), arg);
        6:   23:    arg[s.size()] = '\0';
        6:   24:    args.push_back(arg);
        -:   25:}
        3:   26:args.push_back(nullptr);
        -:   27:
        -:   28:// Fork a process to execute the command
        3:   29:pid_t pid = fork();
        3:   30:if (pid == 0) {
        -:   31:    // Child process
        1:   32:    execvp(args[0], args.data());
        -:   33:    // If execvp returns, an error occurred
        1:   34:    perror("execvp");
        1:   35:    exit(1);
        2:   36:} else if (pid > 0) {
        -:   37:    // Parent process
        -:   38:    int status;
        2:   39:    waitpid(pid, &status, 0);
```

```
    2:   39:    waitpid(pid, &status, 0);
    2:   40:    if (WIFEXITED(status)) {
    2:   41:        int exit_status = WEXITSTATUS(status);
    2:   42:        std::cout << "Program exited with status: " << exit_status << std::endl;
    -:   43:    }
    -:   44:} else {
    -:   45:    // Fork failed
#####:   46:    perror("fork");
#####:   47:    exit(1);
    -:   48:}
    -:   49:
    -:   50:// Free allocated memory
    8:   51:for (char* arg : args) {
    6:   52:    delete[] arg;
    -:   53:}
    2:   54:}
    -:   55:
    4:   56:int main(int argc, char* argv[]) {
    4:   57:if (argc != 3 || std::string(argv[1]) != "-e") {
    1:   58:    std::cerr << "Usage: mync -e \"ttt + (9 different digits as argument)\"\n";
    1:   59:    return 1;
    -:   60:}
    -:   61:
    3:   62:std::string command = argv[2];
    3:   63:executeProgram(command);
    -:   64:
    2:   65:return 0;
    2:   66:}
tik@mastik-VirtualBox:~/ComputerScience/OperatingSystems/Ex2/prog2$
```

תרגיל 3

```
mastik@mastik-VirtualBox:~/ComputerScience/OperatingSystems/Ex2/prog3$ cat mync.cpp.gcov
        -:    0:Source:mync.cpp
        -:    0:Graph:mync.gcno
        -:    0:Data:mync.gcda
        -:    0:Runs:9
        -:    1:#include <iostream>
        -:    2:#include <sstream>
        -:    3:#include <vector>
        -:    4:#include <cstring>
        -:    5:#include <unistd.h>
        -:    6:#include <cstdlib>
        -:    7:#include <sys/wait.h>
        -:    8:#include <sys/types.h>
        -:    9:#include <sys/socket.h>
        -:   10:#include <netinet/in.h>
        -:   11:#include <arpa/inet.h>
        -:   12:#include <netdb.h>
        -:   13:
        -:   14:using namespace std;
        -:   15:
        -:   16:
        7:   17:int TCPServer(int port, bool handle_output = false) {
        -:   18:int server_fd, new_socket;
        -:   19:struct sockaddr_in address;
        7:   20:int opt = 1;
        7:   21:int addrlen = sizeof(address);
        -:   22:
        7:   23:cout << "Starting TCP server on port " << port << endl;
        -:   24:
        7:   25:if ((server_fd = socket(AF_INET, SOCK_STREAM, 0)) == 0) {
    #####:   26:    perror("socket failed");
    #####:   27:    exit(EXIT_FAILURE);
        -:   28:}
        -:   29:
        7:   30:if (setsockopt(server_fd, SOL_SOCKET, SO_REUSEADDR | SO_REUSEPORT, &opt, sizeof(opt))) {
    #####:   31:    perror("setsockopt");
    #####:   32:    exit(EXIT_FAILURE);
        -:   33:}
        -:   34:
        7:   35:address.sin_family = AF_INET;
        7:   36:address.sin_addr.s_addr = INADDR_ANY;
        7:   37:address.sin_port = htons(port);
        -:   38:
        7:   39:if (bind(server_fd, (struct sockaddr *)&address, sizeof(address)) < 0) {
    #####:   40:    perror("bind failed");
    #####:   41:    exit(EXIT_FAILURE);
        -:   42:}
```

```
        -:   42:}
        -:   43:
        7:   44:if (listen(server_fd, 3) < 0) {
    #####:   45:    perror("listen");
    #####:   46:    exit(EXIT_FAILURE);
        -:   47:}
        -:   48:
        7:   49:if ((new_socket = accept(server_fd, (struct sockaddr *)&address, (socklen_t*)&addrlen)) <
    #####:   50:    perror("accept");
    #####:   51:    exit(EXIT_FAILURE);
        -:   52:}
        -:   53:
        7:   54:cout << "Accepted connection on port " << port << endl;
        -:   55:
        7:   56:if (handle_output) {
        2:   57:    if (dup2(new_socket, STDOUT_FILENO) < 0) {
    #####:   58:        perror("dup2 output failed");
    #####:   59:        exit(EXIT_FAILURE);
        -:   60:    }
        2:   61:    if (dup2(new_socket, STDERR_FILENO) < 0) {
    #####:   62:        perror("dup2 error output failed");
    #####:   63:        exit(EXIT_FAILURE);
        -:   64:    }
        2:   65:    cout << "Output redirected to socket" << endl;
        -:   66:}
        -:   67:
        7:   68:return new_socket;
        -:   69:}
        -:   70:
        1:   71:int TCPClient(const string& hostname, int port) {
        -:   72:struct sockaddr_in serv_addr;
        -:   73:struct hostent *server;
        1:   74:int sock = socket(AF_INET, SOCK_STREAM, 0);
        1:   75:if (sock < 0) {
    #####:   76:    perror("ERROR with opening socket..");
    #####:   77:    exit(EXIT_FAILURE);
        -:   78:}
        1:   79:server = gethostbyname(hostname.c_str());
        1:   80:if (server == nullptr) {
    #####:   81:    fprintf(stderr, "ERROR, no such host\n");
    #####:   82:    exit(EXIT_FAILURE);
        -:   83:}
        1:   84:bzero((char *) &serv_addr, sizeof(serv_addr));
        1:   85:serv_addr.sin_family = AF_INET;
        1:   86:bcopy((char *)server->h_addr, (char *)&serv_addr.sin_addr.s_addr, server->h_length);
        1:   87:serv_addr.sin_port = htons(port);
        -:   88:
```

Show Applications

```
    -:   88:
    1:   89:cout << "Connecting to " << hostname << " on port " << port << endl;
    -:   90:
    1:   91:if (connect(sock, (struct sockaddr *)&serv_addr, sizeof(serv_addr)) < 0) {
#####:   92:    perror("ERROR with connecting..");
#####:   93:    exit(EXIT_FAILURE);
    -:   94:}
    -:   95:
    1:   96:cout << "Connected to " << hostname << " on port " << port << endl;
    -:   97:
    1:   98:return sock;
    -:   99:}
    -:  100:
    1:  101:void redirect_IO(int input_fd, int output_fd) {
    1:  102:cout << "Redirecting input and output.." << endl;
    -:  103:
    1:  104:if (input_fd != STDIN_FILENO) {
    1:  105:    if (dup2(input_fd, STDIN_FILENO) < 0) {
#####:  106:        perror("dup2 input failed..");
#####:  107:        exit(EXIT_FAILURE);
    -:  108:    }
    -:  109:}
    -:  110:
    1:  111:if (output_fd != STDOUT_FILENO) {
#####:  112:    if (dup2(output_fd, STDOUT_FILENO) < 0) {
#####:  113:        perror("dup2 output failed..");
#####:  114:        exit(EXIT_FAILURE);
    -:  115:    }
#####:  116:    if (dup2(output_fd, STDERR_FILENO) < 0) {
#####:  117:        perror("dup2 error output failed..");
#####:  118:        exit(EXIT_FAILURE);
    -:  119:    }
    -:  120:}
    -:  121:
    1:  122:cout << "Input and output redirected" << endl;
    1:  123:}
    -:  124:
    1:  125:void printError(const string& msg) {
    1:  126:cerr << msg << endl;
    1:  127:exit(EXIT_FAILURE);
    -:  128:}
    -:  129:
    -:  130:
    9:  131:int main(int argc, char* argv[]) {
    -:  132:int opt;
    9:  133:string command;
    9:  134:string input_source, output_dest;
    -:  135:
   26:  136:while ((opt = getopt(argc, argv, "e:i:o:b:")) != -1) {
   17:  137:    switch (opt) {
    8:  138:        case 'e':
    8:  139:            command = optarg;
    8:  140:            break;
    6:  141:        case 'i':
    6:  142:            input_source = optarg;
    6:  143:            break;
    1:  144:        case 'o':
```

```
      1:  144:        case 'o':
      1:  145:            output_dest = optarg;
      1:  146:            break;
      2:  147:        case 'b':
      2:  148:            input_source = optarg;
      2:  149:            output_dest = optarg;
      2:  150:            break;
  #####:  151:        default:
  #####:  152:            printError("Usage: mync -e \"command\" [-i TCPS<PORT> | -o TCPC<IP,PORT> | -b TCPS<PORT>]");
      -:  153:    }
      -:  154:}
      -:  155:
      9:  156:if (command.empty()) {
     1*:  157:    printError("Usage: mync -e \"command\" [-i TCPS<PORT> | -o TCPC<IP,PORT> | -b TCPS<PORT>]");
      -:  158:}
      -:  159:
      8:  160:cout << "Command: " << command << endl;
      8:  161:cout << "Input source: " << input_source << endl;
      8:  162:cout << "Output destination: " << output_dest << endl;
      -:  163:
     15:  164:stringstream ss(command);
     15:  165:string program;
     15:  166:vector <string> args;
      8:  167:ss >> program;
     15:  168:string arg;
     14:  169:while (ss >> arg) {
      6:  170:    args.push_back(arg);
      -:  171:}
      -:  172:
     15:  173:vector<char*> exec_args;
      8:  174:exec_args.push_back(strdup(program.c_str()));
     14:  175:for (const auto& a : args) {
      6:  176:    exec_args.push_back(strdup(a.c_str()));
      -:  177:}
      8:  178:exec_args.push_back(nullptr);
      -:  179:
      -:  180:
      8:  181:cout << "Executing program: " << program << endl;
      8:  182:cout << "Arguments: ";
     14:  183:for (const auto& a : args) {
      6:  184:    cout << a << " ";
      -:  185:}
      8:  186:cout << endl;
      -:  187:
      8:  188:int input_fd = STDIN_FILENO;
      8:  189:int output_fd = STDOUT_FILENO;
      -:  190:
      8:  191:if (!input_source.empty()) {
      7:  192:    if (input_source.substr(0, 4) == "TCPS") {
      7:  193:        int port = stoi(input_source.substr(4));
      7:  194:        bool handle_output = (input_source == output_dest);
      7:  195:        input_fd = TCPServer(port, handle_output);
      7:  196:        if (handle_output) {
      2:  197:            output_fd = input_fd;
                    }
      -:  200:}
```

Show Applications

```
        -:  200:}
        -:  201:
        8:  202:if (!output_dest.empty()) {
        3:  203:    if (output_dest.substr(0, 4) == "TCPC") {
        1:  204:        size_t comma_pos = output_dest.find(',');
        1:  205:        string hostname = output_dest.substr(4, comma_pos - 4);
        1:  206:        int port = stoi(output_dest.substr(comma_pos + 1));
        1:  207:        output_fd = TCPClient(hostname, port);
       3*:  208:    } else if (output_dest.substr(0, 4) == "TCPS" && output_dest != input_source) {
    #####:  209:        int port = stoi(output_dest.substr(4));
    #####:  210:        output_fd = TCPServer(port, true);
        -:  211:    }
        -:  212:}
        -:  213:
        8:  214:pid_t pid = fork();
        8:  215:if (pid < 0) {
    #####:  216:    perror("fork");
    #####:  217:    return EXIT_FAILURE;
        8:  218:} else if (pid == 0) {
        1:  219:    cout << "In child process, executing command" << endl;
        1:  220:    redirect_IO(input_fd, output_fd);
        1:  221:    execv(exec_args[0], exec_args.data());
        1:  222:    perror("execv");
        1:  223:    exit(EXIT_FAILURE);
        -:  224:} else {
        -:  225:    int status;
        7:  226:    cout << "wait for child to finish.." << endl;
        7:  227:    waitpid(pid, &status, 0);
        7:  228:    if (WIFEXITED(status)) {
        7:  229:        cout << "Child process exited with status " << WEXITSTATUS(status) << endl;
        -:  230:    } else {
    #####:  231:        cout << "Child process did not exit successfully" << endl;
        -:  232:    }
        -:  233:}
        -:  234:
       27:  235:for (auto arg : exec_args) {
       20:  236:    free(arg);
        -:  237:}
        -:  238:
        7:  239:return 0;
```

Show Applications

```
mastik@mastik-VirtualBox:~/ComputerScience/OperatingSystems/Ex2/prog3$
```