ΧΑΡΟΚΟΠΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ & ΤΗΛΕΜΑΤΙΚΗΣ

# Κατσιφώλης Βασίλης

2η Εργασία στο μάθημα **Λειτουργικά Συστήματα**

Ταύρος, 30 Ιανουαρίου 2019

Περιεχόμενα

# Άσκηση remote shell

## Server

### Κώδικας

Το πρόγραμμα που δημιουργήθηκε μαζί με τα σχόλια είναι:

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

/* Macros */
#define QUIT 127
#define BUFFER_SIZE 2048
#define printb(...)	printf("\033[32m" __VA_ARGS__)
#define printr(...)	printf("\033[31m" __VA_ARGS__)
#define printm(...)	printf("\033[36m" __VA_ARGS__)
#define COLOR_RESET	printf("\033[0m")

/* Function Declarations */
void error(const char *);
void sig_int(int);
void check_child_exit(int);
char parse(char *[], char *);
void runpipe(int [], int, char *);

/* Globals */
int game_counter;

void
error(const char *msg)
{
	perror(msg);
	exit(1);
}

void
sig_int(int signum)
{
	printf("\nCaught signal\nNUMBER OF SIGNAL: %d ",signum);
	exit(signum);
}

void
```

```
check_child_exit(int status)
{
	if(WIFEXITED(status)) {
		printf("Child ended normally. Exit code is %d\n",WEXITSTATUS(status));
	} else if (WIFSTOPPED(status)) {
		printf("Child ended because of an uncaught signal, signal =
%d\n",WTERMSIG(status));
	} else if (WIFSTOPPED(status)) {
		printf("Child process has stopped, signal code = %d\n",WSTOPSIG(status));
	}
	exit(EXIT_SUCCESS);
}

/* Parses the command sent by the client */
char
parse(char *vec[10], char *line)
{
	int i;
	char *pch = malloc(sizeof(line));
	if(strstr(line,"|")) {
		printf("Found pipe\n");
		pch = strtok(line, "|");
		i=0;
		while (pch != NULL) {
			vec[i]=pch;
			pch = strtok (NULL, "|");
			i++;
		}
	} else {

		pch = strtok (line," \n");
		i=0;
		while (pch != NULL) {
			vec[i]=pch;
			pch = strtok (NULL, " \n");
			i++;
		}

		vec[i]='\0';
	}
	printf("%s\n", vec);

	return 0;
}
```

```
void
runpipe(int pfd[],int socket,char *typedcommand)
{
	char* vec[10];
	char* comands[3];
	parse(comands,typedcommand);
	int pid;
	switch(pid=fork()){
		/* child */
		case 0:
			/* Parsing, duplicating the fds for in, out, err and executing */
			parse(vec,comands[1]);
			dup2(pfd[0],0);
			dup2(socket,1);
			dup2(socket,2);
			close(pfd[1]);
			/* the child does not need this end of the pipe */
			execvp(vec[0],vec);
			perror(vec[0]);
			exit(1);

		/* parent */
		default:
			parse(vec,comands[0]);
			dup2(pfd[1],1);
			close(pfd[0]);
			execvp(vec[0],vec);
			perror(vec[0]);
			exit(1);
		case -1:
			perror("fork");
			exit(1);
	}
}

void
game(int counter)
{
	int i;
	int holder[counter];
	printf("Please pick a number from 0 - 20 %d times: ", counter);
	for (i = 0; i < counter; i++) {
		printf("Pick your number: ");
		scanf("%d", holder[i]);
	}
```

```
}

int
main(int argc, char *argv[])
{
	int sockfd, newsockfd, portno;
	socklen_t clilen;
	char buffer[BUFFER_SIZE];
	struct sockaddr_in serv_addr, cli_addr;
	int n;
	char str[INET_ADDRSTRLEN];
	pid_t childpid;
	int pid,status;
	char *vec[10];
	size_t bufsize = 32;
	size_t characters;

	/* Signal Contoller */
	signal(SIGINT,sig_int);

	if (argc < 2) {
		fprintf(stderr, "No port provided\n");
		exit(1);
	}

	/* Socket Creation */
	if ((sockfd = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
		error("ERROR opening socket");
	}
	printf("Server socket is created.\n");


	memset((char *)&serv_addr, '\0', sizeof(serv_addr));
	portno = atoi(argv[1]);
	serv_addr.sin_family = AF_INET;
	serv_addr.sin_addr.s_addr = INADDR_ANY;
	serv_addr.sin_port = htons(portno);

	/* Binding */
	if (bind(sockfd,(struct sockaddr *) &serv_addr, sizeof(serv_addr)) < 0) {
		error("ERROR on binding");
	}

	printf("Bind to port %d\n",portno);
```

```
	/* Listening */
	listen(sockfd, 5);
	printf("Listening for connections..\n");

	while (1) {
		/* Wait for incoming connections in a loop */

		clilen = sizeof(cli_addr);
		newsockfd = accept(sockfd, (struct sockaddr *) &cli_addr, &clilen);

		if (newsockfd < 0) {
			error("ERROR on accept");
		}
		printf("Connection accepted\n");


		if (inet_ntop(AF_INET, &cli_addr.sin_addr, str, INET_ADDRSTRLEN) ==
NULL) {
			fprintf(stderr, "Could not convert byte to address\n");
			exit(1);
		}
		printf("The client address is :%s\n", str);


		/* Fork the process to poll for commands */
		if((childpid = fork()) == 0 ) {
			while(1) {
				printf("Client %s please enter a command\n",str);

				memset(buffer, '\0', BUFFER_SIZE);
				n = recv(newsockfd, buffer, BUFFER_SIZE - 1, 0);
				if (n <= 0) {
					error("client disconnected");
				} else if (n == '\n') {
					continue;
				}

				int pid=fork(); /* Fork to execute the specified command */
				if(pid==-1) {
					perror("fork");
					exit(1);
				}
				if(pid!=0) {
					/*
```

```
                                        * Waiting for the child to return
                                        * and kill it
                                        */
                                       if(wait(&status)==-1) {
                                                perror("wait");
                                                check_child_exit(QUIT);
                                       }

                                  } else {
                                                /* Client disconnects and server closes the socket */
                                                if(strcmp(buffer, "END\n") == 0 || n < 0 ) {
                                                        fprintf(stdout,"Disconnected from client %s\n",str);
                                                        close(sockfd);
                                                        exit(QUIT);
                                                        }
                                                        printf("Executing Client's %s command:",str);

                                                        /* pipe */
                                                        if(strstr(buffer,"|")) {
                                                                int pid;
                                                                int fd[2];
                                                                pipe(fd);
                                                                switch(pid=fork()){
                                                                        case 0:

runpipe(fd,newsockfd,buffer);
                                                                        default:

while((pid=wait(&status))!=-1);
                                                                                exit(0);
                                                                        case -1:
                                                                                perror("fork");
                                                                                exit(1);

                                                                }

                                                         } else {
                                                                parse(vec,buffer);
                                                                dup2(newsockfd,1);
                                                                dup2(newsockfd,2);
                                                                execvp(vec[0],vec);
                                                                perror(vec[0]);
                                                                exit(1);
                                                        }
```

```
                                        }
                                }

                        }

                }
        close(newsockfd);

        return 0;
}
```

## Client

### Κώδικας

Το πρόγραμμα που δημιουργήθηκε μαζί με τα σχόλια είναι:

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

#define BUFFER_SIZE 2048
#define GAME_COUNTER 20

/* Macros */

#define COLOR_RESET       printf("\033[0m")
#define printb(...)       printf("\033[32m" __VA_ARGS__)
#define printr(...)       printf("\033[31m" __VA_ARGS__)
#define printm(...)       printf("\033[36m" __VA_ARGS__)

/* Function Declarations */
void error(const char *);
void sig_int(int);
void check_server(int);
char * game(int);


/* Globals */
int client_socket;
int game_counter;

void
error(const char *msg)
{
        fprintf(stderr, msg);
        exit(0);
}

void
sig_int(int signum)
```

```
{
	printf("\nCaught signal %d \n",signum);

	exit(signum);
}

void
check_server(int signum)
{
	char tmp[100];
	memset(tmp, '\0', sizeof(tmp));
	if (recv(client_socket, tmp, 100, MSG_DONTWAIT) == 0) {
			close(client_socket);
			error("Server Shutdown\n");
			exit(1);
	}
	return;
}

char *
game(int counter)
{
	int i;
	char tmp;
	char *game_vec = malloc(sizeof(1000));
	printf("Please pick a number from 0 - 20 %d times\n", counter);
	for (i = 0; i < counter; i++) {
		tmp = 0;
		printf("Pick your number: ");
		scanf(" %c", &tmp);
		game_vec[i] = tmp;
	}
	game_vec[++i] = '\0';

	return game_vec;

}

int
main(int argc, char *argv[])
{
	/* Initializations */
	srand(time(NULL));
	int result,portno;
	struct sockaddr_in serverAddr;
```

```
char buffer[BUFFER_SIZE];
char game_vec[GAME_COUNTER];
int readbytes;
char tmp[100];

signal(SIGINT, sig_int);
signal(SIGALRM, check_server);

/*
 * Sets an alarm
 * to check every (interval)
 * if the server has shut-down
 */
struct itimerval timer;
/* First expiration */
timer.it_value.tv_sec = 0;
timer.it_value.tv_usec = 500000;
/* Interval (every then) */
timer.it_interval.tv_sec = 0;
timer.it_interval.tv_usec = 50000; /* in ms */
setitimer (ITIMER_REAL, &timer, NULL);

/* Guard */
if (argc < 3)
{
        fprintf(stderr, "usage %s ip-address port\n", argv[0]);
        exit(0);
}

portno = atoi(argv[2]);
/* Creates the socket */
client_socket = socket(AF_INET, SOCK_STREAM, 0);
if (client_socket < 0) {
        error("ERROR opening socket");
}
printb("Client socket is created.\n");
COLOR_RESET;

memset(&serverAddr, '\0', sizeof(serverAddr));

/* Populating the serverAddr struct */
serverAddr.sin_family = AF_INET;
serverAddr.sin_addr.s_addr=inet_addr(argv[1]);
serverAddr.sin_port = htons(portno);
```

```
	/* Finally conect to the server */
	result = connect(client_socket, (struct sockaddr *) &serverAddr, sizeof(serverAddr));
	if (result < 0) {
		error("ERROR connecting");
	}

	printb("Connected to server with ip address: \n");
	COLOR_RESET;

	/* main loop for the client */
	for (;;) {
		printm("%s:>",argv[1]);
		COLOR_RESET;
		memset(buffer, '\0', BUFFER_SIZE); /* fill the buffer with 0 */
		fgets(buffer, BUFFER_SIZE - 1, stdin); /* get input */
		send(client_socket, buffer, strlen(buffer), 0);
		if (strcmp(buffer, "END\n") == 0) {
			close(client_socket);
			printf("Disconnected from server %s \n",argv[1]);
			break;
		}
		if (strcmp(buffer, "\n") == 0) continue;

		/* blocks until a response comes from the server */
		readbytes = recv(client_socket, buffer, 7000, 0);
		if (readbytes == 0) {
			close(client_socket);
			error("Server Shutdown\n");
			break;
		} else {
			game_counter++; /* Incrementing for each successful command */
			buffer[readbytes-1]='\0';
			printr("Server:\n%s%d\n", buffer, game_counter);
			COLOR_RESET;
		}
	}
	return 0;
}
```

Ενδεικτικές εκτελέσεις (screenshots) ανα περίπτωση:

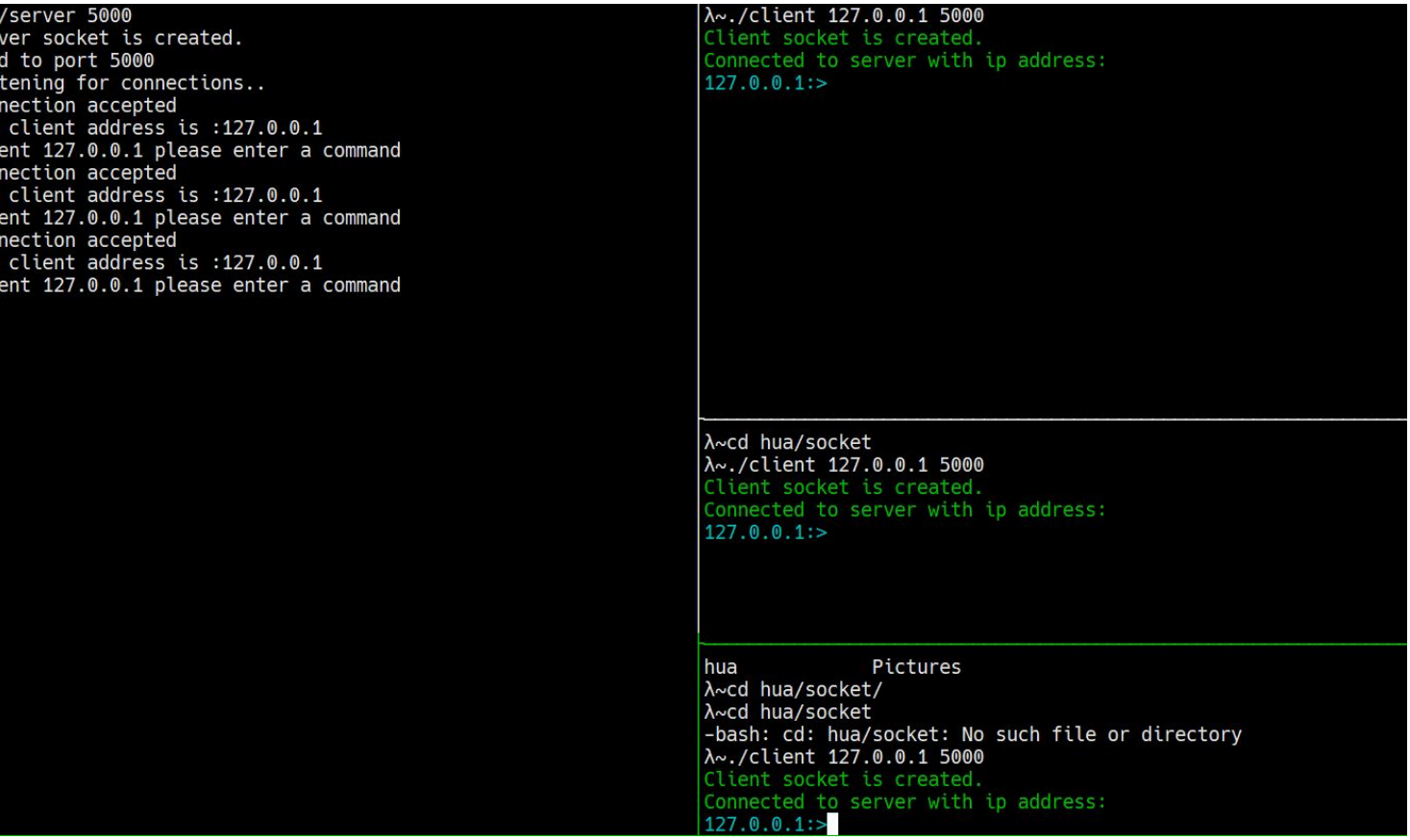
- **client-server επικοινωνία**


*Παρατηρήσεις:*

- **Ομαλή λειτουργία client server**

*Παρατηρήσεις:*

- **υποστήριξη πολλών client**

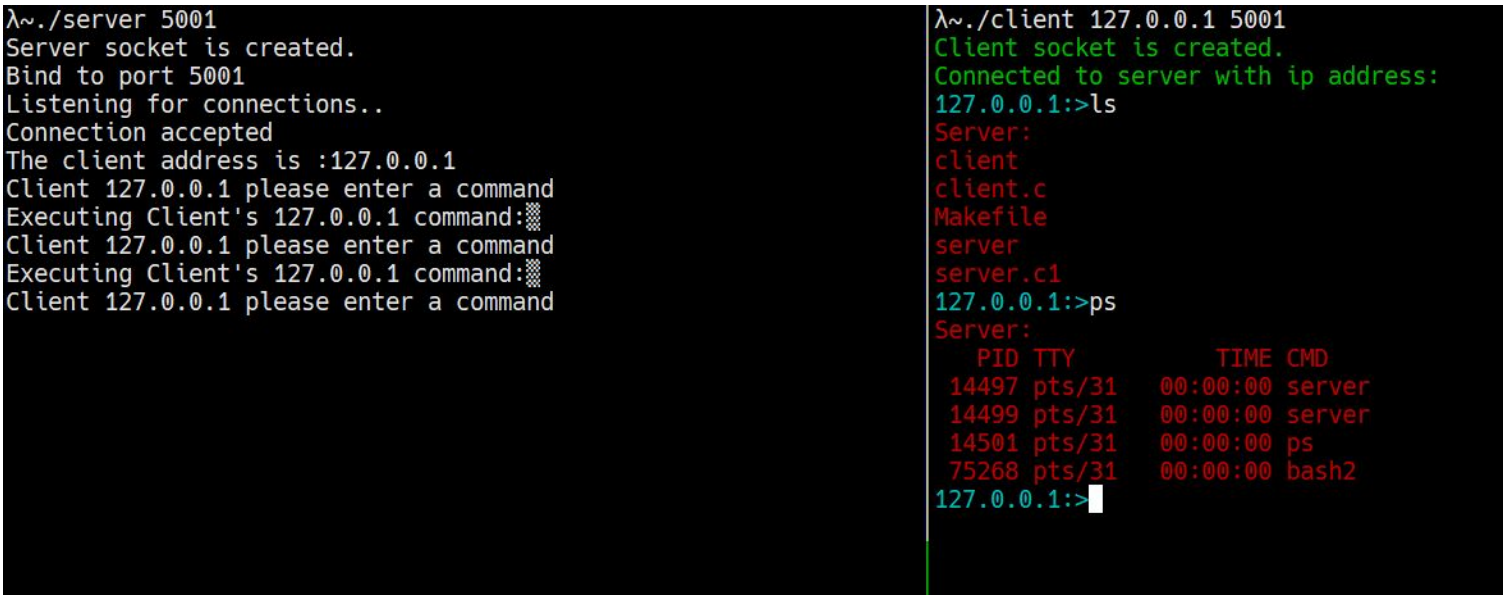



*Παρατηρήσεις:*

- **Υποστήριξη απλών εντολών**

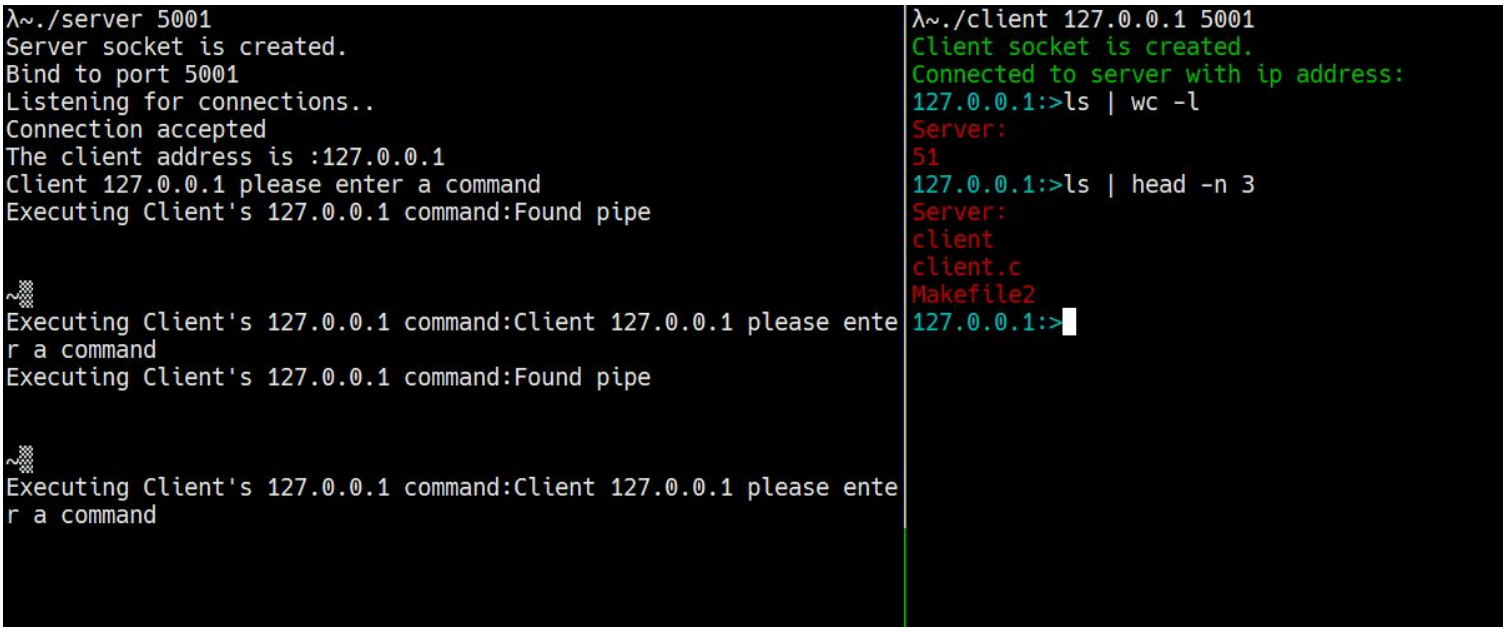
*Παρατηρήσεις:*

- **Υποστήριξη εντολών με παραμέτρους και ορίσματα**




*Παρατηρήσεις:*

- **Υποστήριξη απλών σωληνώσεων**



*Παρατηρήσεις:*

## Γενικά Σχόλια/Παρατηρήσεις

Όταν ο client στέλνει μια εντολή στον server για parsing εμφανίζεται στην οθόνη του server η εντολή που εκτελέστηκε αλλά μου εμφανίζει την εντολή με περίεργη κωδικοποίηση.

## Με δυσκόλεψε / δεν υλοποίησα

Με δυσκόλεψε να στείλω στον server τους τυχαιούς αριθμούς του παιχνιδιού μέσω του socket. Όταν προσπαθούσα να στείλω τον buffer με την send το errno έπαιρνε την τιμή 1 = EPERM και η recv του server έπαιρνε τιμή 0 και έπειτα ο client τερμάτιζε.

# Συνοπτικός Πίνακας

| 2η Άσκηση | | |
|---|---|---|
| | Υλοποιήθηκε **(ΝΑΙ/ΟΧΙ/ΜΕΡΙΚΩΣ)** | **Παρατηρήσεις** |
| client-server επικοινωνία | ΝΑΙ | |
| Ομαλή λειτουργία client server | ΝΑΙ | |
| υποστήριξη πολλών client | ΝΑΙ | |
| Υποστήριξη απλών εντολών | ΝΑΙ | |
| Υποστήριξη εντολών με παραμέτρους και ορίσματα | ΝΑΙ | |

| | | |
|---|---|---|
| Υποστήριξη απλών σωληνώσεων | ΝΑΙ | |