Lab 2 OS

-lab test in c

-how to send inputs, from a file, from keyboard (60%), dynamic allocating memory( ex: matrix )

-(40%) algorithm - read from file a matrix + do something with it

NANO -expects a file name!!

nano test.c

^ ctrl

^s ctrl s(save)

^x ctrl x(exit terminal)

First libraries!!

Body of program:

Main function – executed first, entry point – expects an integer !!only on an operating system,

Specific signature: 2 arguments: have a program that receives 2 arguments

-through arguments-send to the executable, make it dynamic in s

ARGS(argument count) – integer, automatically populated during the program, receives arguments

ARGV(argument vector) – define from scratch, pointer!!, no list, vector, hand-crafted, a block of small data types, collection of a type, always a pointer that points at the first element of the array

=list of strings always!!= array of characters

Ex: ana are mere => 3 arrays:

A double pointer of type character (\* at the beginning of everything)

\*

\*a n a

\*a r e

\*m e r e

Return 0 – success!!!- at the end

Return 1 – fail/error

Valgrind!!!!!!!!!!!!!!!

Text

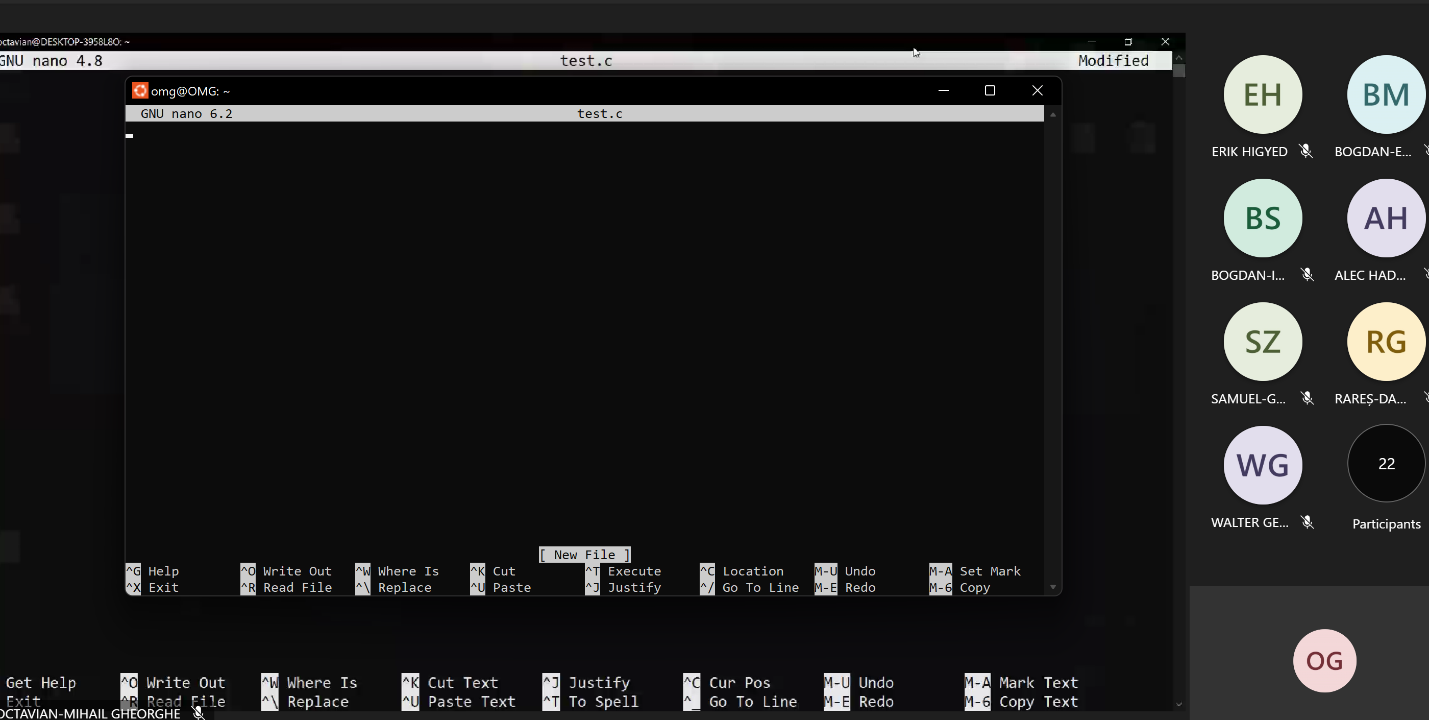
Description automatically generated

Dupa ce scri

Ctrl s

Ctrl x

Apoi cat part 😊



Cat -n test.c

Print the file with an endl

Gcc -Wall test.c -o test.exe- to compile with warnings shown+name file + output

!!pay attention to not have the same name(test.c != test.exe)!!

./ test.exe (execute something – the executable, not the c program!!)

Gcc install

Sudo apt-get update

Sudo apt install gcc

Up arrow – to navigate previous commands

./test.exe Ana are 3 mere =>5 is the output

Scanf( “%d”, &n)

Type, reference of a variable – store smth into a variable through a function,

& = passing the address memory of the variable (we want to store something!!)

Test: c program with errors

Code – compilation errors, warnings

Run code + solve

Ctrl s

Ctrl x

Gcc -Wall test.c -o test.exe

./test.exe

Type number

Nano matrix.txt

Format: 4X4 matrix, matrix

Cat -n test.c

FILE \*f;

f = open(argv[1],’r’);

int n,m;

fscanf(f,”%d %d”,&n,&m);

nano test.c

matrix = (int\*\*)malloc(n\*sizeof(\*int));

nr of bytes to be allocated

we have n – n pointers

allocate – always deallocate!!

For(i=0;i<n;i++)

Free(matrix[i]);

Free(matrix);

&matrix – reference!!

Matrix – value!!