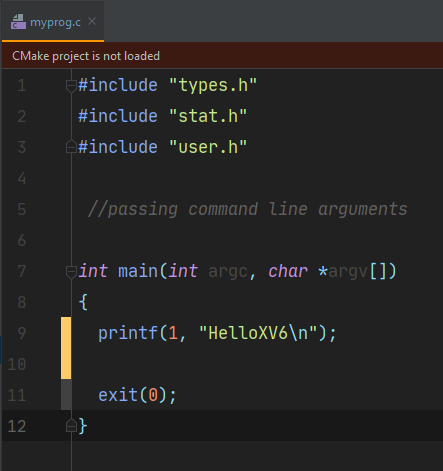
Operating System

Assignment 1

Eran Aflalo 209343722

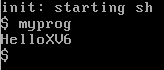
Omer Luxembourg 205500390

**Task 1: Warm up (“HelloXV6”):**

This part of the assignment is aimed at getting you started. It includes a small change in xv6 shell. Note that in terms of writing code, the current xv6 implementation is limited: it does not support system calls you may use when writing on Linux and its standard library is quite limited.

We created the following C program and add it to the Makefile – added myprog.c to ‘EXTRA’ section and \_myprog to ‘UPROG’ section.

Results below:



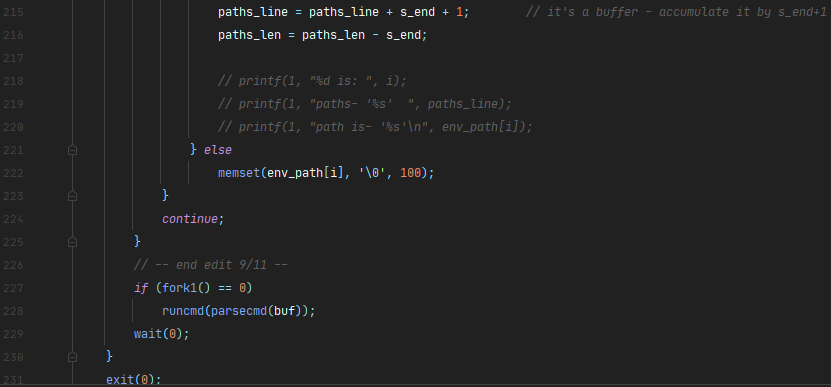
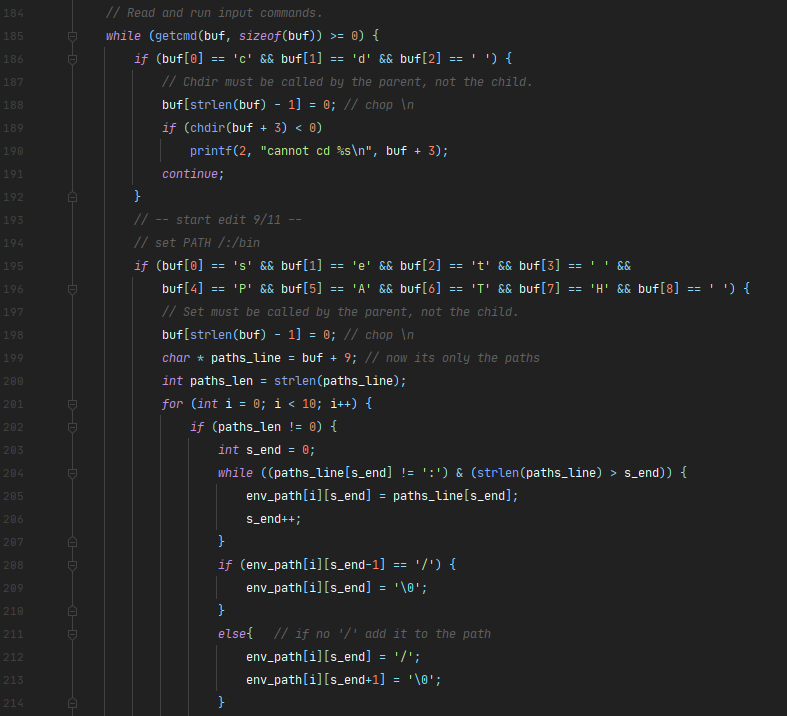
**Task 2: Support the PATH environment variable:**

In this task we edit *sh.c* to maintain an array of 10 paths which are variable length up to 100 chars, that will be the environment paths as we know from Linux.

First, we define a global array for the path:

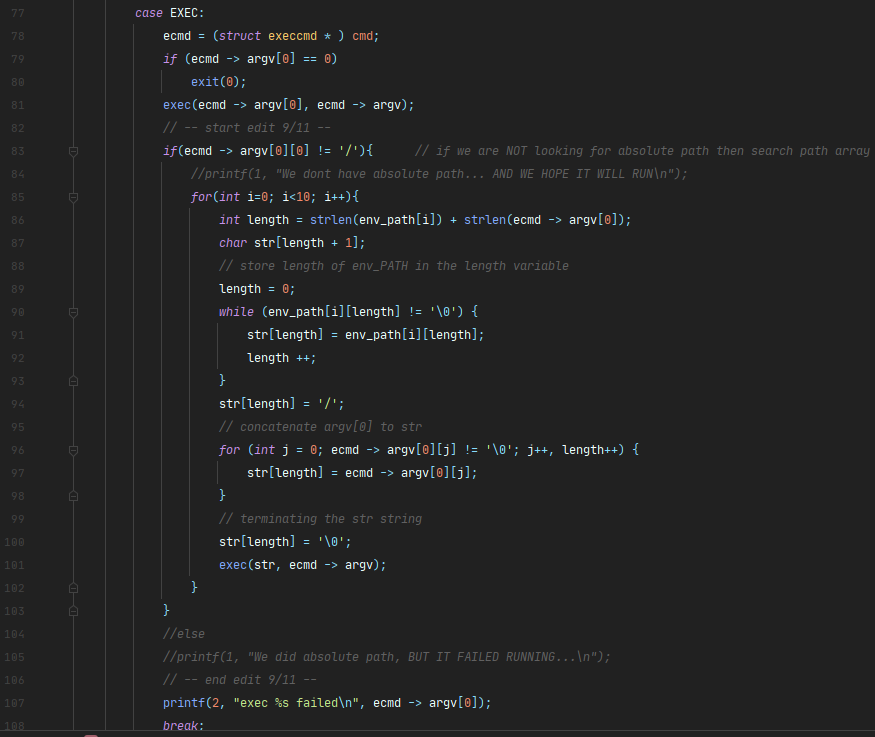
*char* env\_path[10][100];

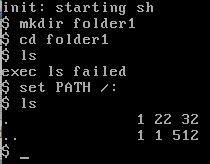
Next we added to the ‘main’ function the following code:



The code will allow us to run the command ‘set PATH’ and afterward the paths we want to add to the environment.

After this edition, we used the paths array to find commands to run from a current path, which are not the ‘*home*’ path (“/”) or order directories, which are not the current directory – the following code is in *sh.h* in the ‘runcmd’ function:



Results below:

Before setting the PATH, we can’t use the *ls* command, because it is in the *home* directory. After we added the *home* directory, which is “/”, we could run the command *ls*.

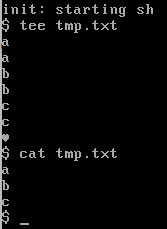
\* note: if we run *“/ls”* for the directory *“/folder1”*, the command will work, as requested.

**Task 3: Extend Functionality of XV6**

**3.1** We added user space program called *tee* as requested.

We created a C program and added it to the Makefile. The whole code is in *tee.c*.

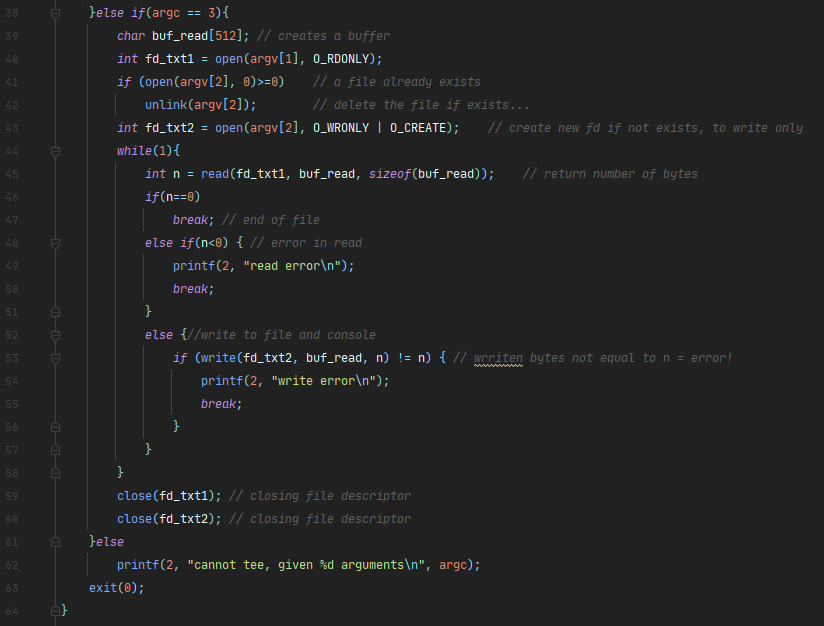
The first section is for a single file:

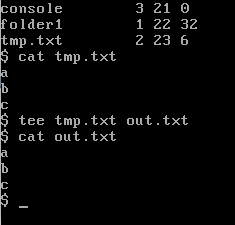


The results of a single file:

Echo after each Enter

This is CTRL+C and then Enter

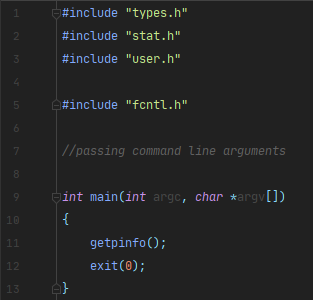
For two files we created the following code:

The results are as follows:

As we can see the content of *tmp.txt* is in *out.txt* now.

**3.2** *getpinfo* is a system call that print a list of currently running processes.

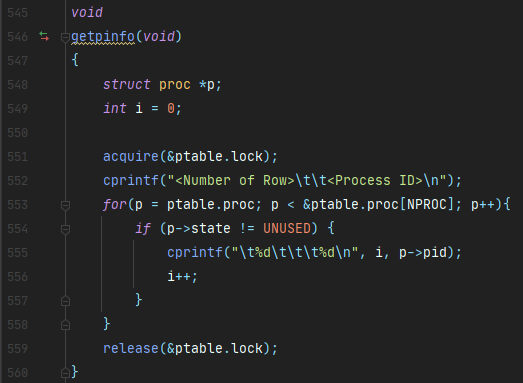
To implement it, we created a user space program and a kernel system call as follows:

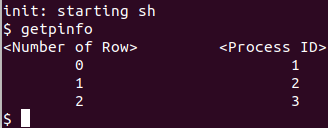
Use space program only calls the system call –

To create the kernel system call we added the following files with appropriate function calls and macros:

*user.h ysys.S syscall.c syscall.h*

In addition, we created the following function to *proc.c* file:

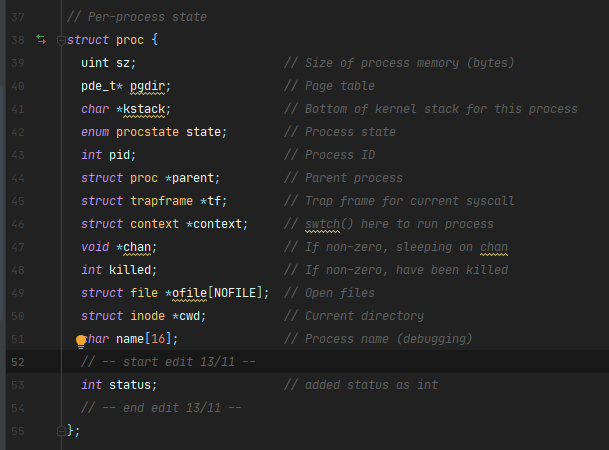


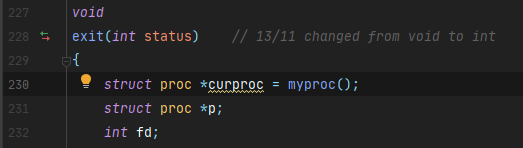
Results:

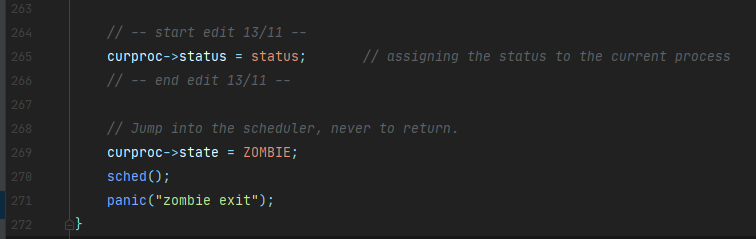
\* Note that the QEMU terminal does not identify “\t” as a regular tab.

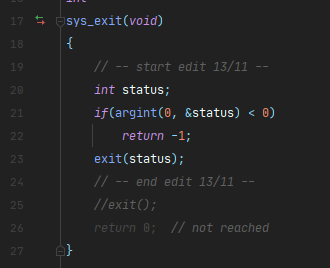
**Task 4: Wait and exit system calls**

**4.1** We updated the process structure with a status field in ‘*proc.h’* (and updated *user.h, defs.h, sysrpoc.c, proc.c*). In addition, we edited *exit* function (in *proc.c*) signature to receive an int status.

*proch.h* – added a field

*proc.c* – exit function

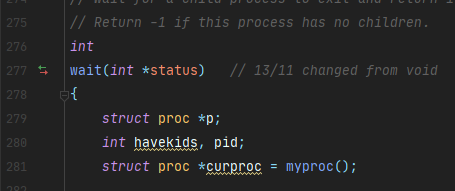
 …

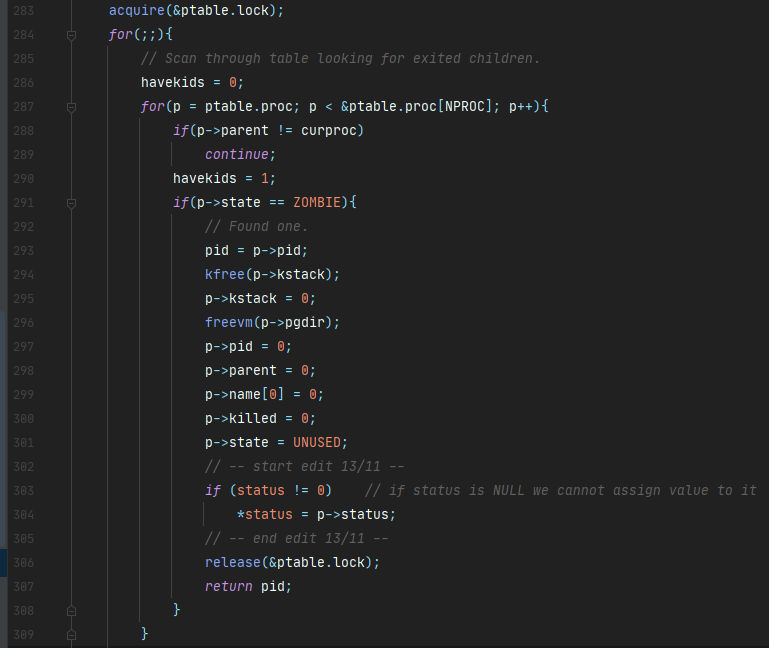
*sysproc.c* –

After changing *exit* function, we changed “*exit()”* to *“exit(0)”* where needed.

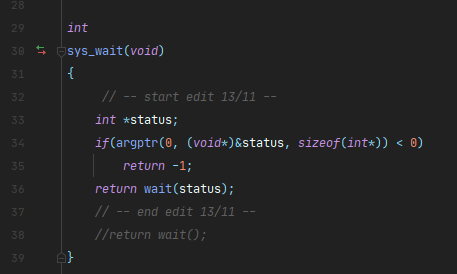
**4.2** Updating the wait system call. We changed *wait* system call to resolve with the status given in exit.

Same as *exit* we updated the files:

*proc.c* -

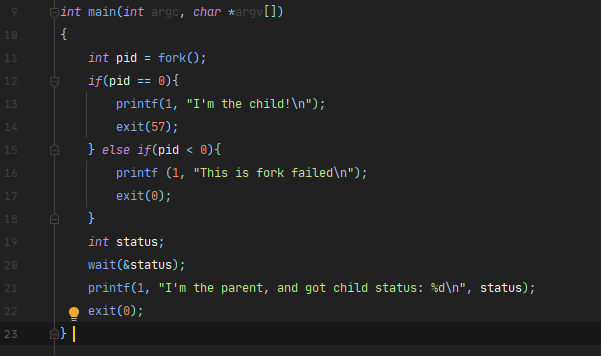


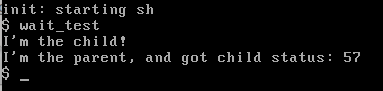
The status in this function is a pointer to status field in ‘p’ process.

*sysproc.c* –

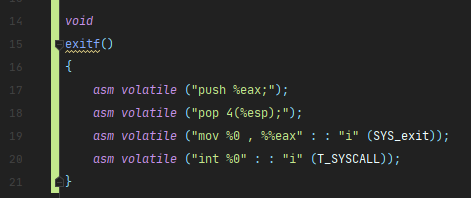
We created a user space program to test the functionality of the status with wait and exit. The following results show the status value that got from the function by calling “*exit(57)*”.

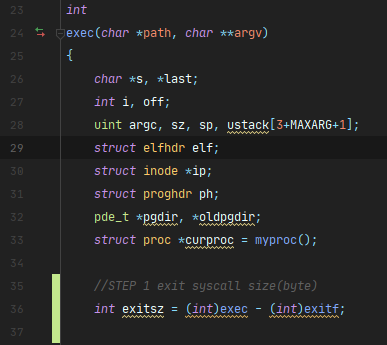
This is the user space program “wait\_test”:

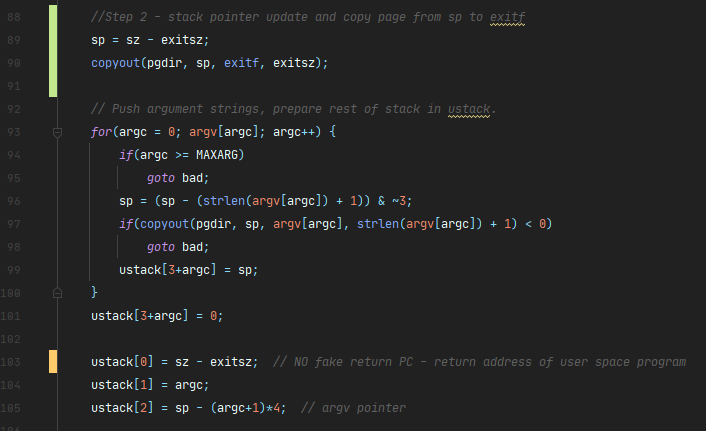


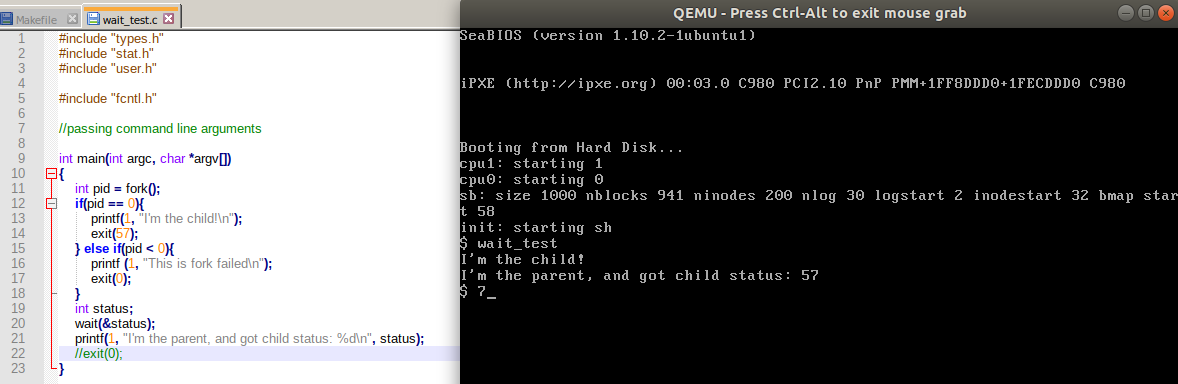
The terminal results:

**4.3** Updating the wait system call – we adjust the “*exec.c*” file as requested.

exitf() – will create the function for the stack.

In the *exec* function we updated the following:



To test it we emitted the “exit(0)” in the user space program “wait\_test.c” that we created before:

As expected, the adjustment worked!