**CSE 3033 - OPERATING SYSTEMS Programming**

**Assignment # 2**

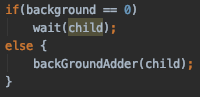
Furkan KUSE - 150117041

Metehan ERTAN – 150117051

A-)

A picture containing text, black

Description automatically generatedIn this part we implemented a function that takes the arguments. Which can be anything that runs on terminal and the ones we created. For running part, we take the all possible paths and add the command name at the end. Then we try to call execv( ) with newly created paths.



In this part the parent process waits for its child if command is not called with &. Otherwise, we add the process to background process.

B-)

In this part the we create some unique commands.

* **A black sign with white text

  Description automatically generatedhistory**

In this part parent process calls library function that adds the used command to the history library linked list.

A screen shot of a smart phone

Description automatically generatedIn this part child process prints the history linked list.

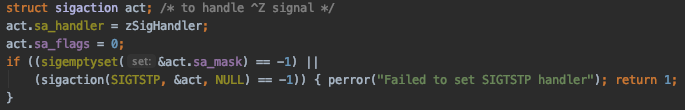
Output of the history command:

A close up of text on a black background

Description automatically generated

We could not implement the history -i part.

* **^z part**

****

In this part we implemented a signal handler. This signal handler works when user presses ctrl-z and calls function zSigHandler.

**A screenshot of a cell phone

Description automatically generated**isForeground and foregroundID are global variables. They are updated when new foreground process enters. ( isForeground becomes 1 and foregroundID becomes the p\_id of newly forked child.)

* **path**

When user enters the path command without any argument, shell process prints current pathname list.

When user enters path command with “+ /path” arguments, shell process adds the given pathname to the current pathname list.

When user enters path command with “- /path” arguments, shell process deletes the given pathname from to the current pathname list if there is given path.

Output of the path part:

A screen shot of a smart phone

Description automatically generatedA close up of text on a black background

Description automatically generated

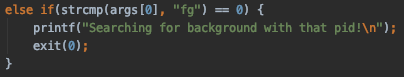
In the pathAdder function, we add the given path to the path linkedlist.

In the pathDelete function, we delete the given path from the path linkedlist with all its duplicate if there is any.

In the pathPrinter function, we print all the paths that are in path linkedlist.

* **fg %num**

When we create a background process, our shell prints p\_id of the newly forked child. With this p\_id and fg command, we can move the background process with given p\_id to foreground if it isn’t died yet.



A black sign with white text

Description automatically generatedWhen we call fg command, newly forked child process will die and parent will call backgroundProcessSearcher function.

A screenshot of a cell phone

Description automatically generated

In this function, program searches for background process with given p\_id. If function finds the child, it checks if that process already died. If that process hasn’t died yet, program stops it using signal. Then, program makes the process continue with another signal and wait for it until it dies.

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

* **exit**

A picture containing object

Description automatically generated

A screenshot of a cell phone

Description automatically generated

When we call exit command, newly forked child process will die, and parent will call waitpid function with given parameters. With given parameters waitpid function returns 0 to report that there are possibly unwaited childs and returns non-zero value when whole children are dead. Our shell calls exit when this function returns non-zero value.

A picture containing text

Description automatically generated

**A screenshot of a cell phone

Description automatically generated** C-)

In C part we were expected to make IO redirection in our shell. There are 4 type of IO redirection. Output writing, output appending, error writing and input directing.

A black and silver text

Description automatically generated

In this part we checked whether there is an output redirection. If there is an output direction this function returns 1 for writing 2 for append and sets output files’ name. Later clears the direction part that is in args for making it acceptable for running.

A screen shot of a computer

Description automatically generated

In this part we checked whether there is an input redirection. If there is an input direction this function returns 1 and sets input files’ name. Later clears the direction part that is in args for making it acceptable for running.

A screen shot of a computer

Description automatically generated

In this part we checked whether there is an error output redirection. If there is an error output direction this function returns 1 and sets error output files’ name. Later clears the direction part that is in args for making it acceptable for running.

A close up of a sign

Description automatically generated

With the turn values from functions we checked which direction is needed than run the needed part.

A screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedHere we opened or created the wanted output file, created the flags, created mode and take the output of this process. With the output of this process we print the error. If there isn’t a problem, we directed the output of the command to given output file. In append all the process is similar.

Here we opened the input file, created the flags, created mode for redirection. With the output of this process checked errors. Later direct this file to the given command as the input.

A screenshot of a cell phone

Description automatically generated

Here we opened or created the wanted error output file, created the flags, created mode and take the output of this process. With the output of this process we print the error. If there isn’t a problem, we directed the error output of the command to given error output file.