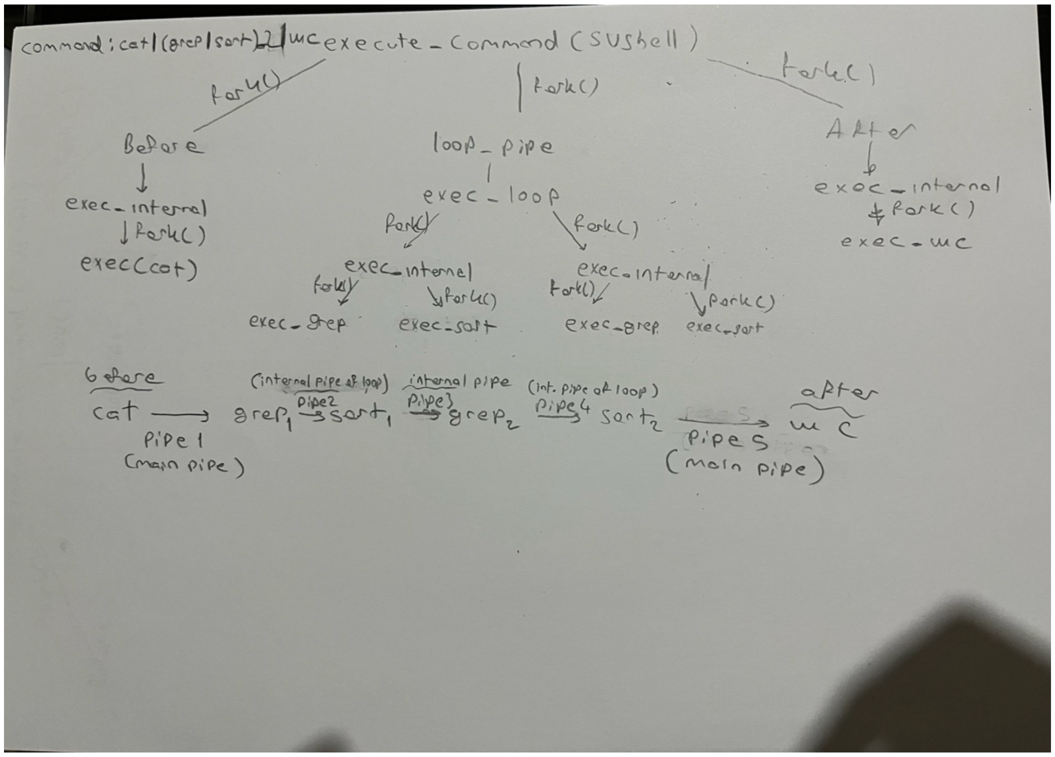
Kaan Berk Karabıyık 34424
CS307 HW 1 REPORT



Process Hierarchy and Concurrency:
exec_command is the main process(SUShell) and it forks the before ,inLoop and after stages that is parsed by the parser.The forked processes execute their helper functions of execute_pipeline_internal or execute_loop_pipe depending on the stage that is determined by the parsed command structure.This forked stages also have their own forks that executes the commands like cat grep etc.

Concurrency:
Exec_command function forks the stages sequentially in the order of the command.For example loop 0 forks the before stage and loop 1 forks the looppipe stage if exists.After the loop starts the forking process one by one and after the processes are forked they continue executing concurrently.The pipes make sure it is sync by its internal concurrenct structure.

Pipe Structure and concurrent access:
There is one pipe per two commands(cat something.txt|wc-l).
The parent exec_command process creates one pipe to connect the stages.

The execute_pipeline_internal function creates additional pipes within a stage if that stage is a pipeline. If it is a single command it does not create a pipeline by checking if it is the last command in command structure.

The later pipe gets the correct input because the parent process that created the pipe saves the read end of that pipe in input_fd_for_current_cmd variable for the next iteration.The next iteration also forks and it calls dup2(input_fd_for_current_cmd, STDIN_FILENO).This makes sure that the STDIN for that fork is the read end of former pipe that is p[0].

Concurrency:
The design of pipes takes care of the concurrency, I did not add any mutexes. If there is an empty pipe it blocks the process that wants to read from it and puts it to sleep. Additionaly if it wants to write to a full pipe it also blocks the process that wants to write.

LoopPipe Input Sources

The LoopPipe has its own STDIN and STDOUT. That STDIN and STDOUT is replaced in exec_command using the dup2 function.
The input source can be:
STDIN(if the C.before part is empty)

A file(if the C.Before part is empty but it is redirectioned to a file using <)

A pipe(if there is C.before part it reads from the read end of that pipe that was saved to a variable by the parent process.)

The program differentiates these cases by using input_fd_for_current_cmd variable.It can be initialized to final_input_fd if there is no C.before part or if there was a C.before part the plumbing logic of the parent updates input_fd_for_current_cmd to be the read-end of the pipe from that stage.

Input for the after Command

This is no different than the same plumbing logic used by the other stages.If it is not the last stage the parent creates a pipe to connect the stages.The parent makes sure the read end of that pipe from the current loop is saved.The after child that is forked calls dup2(input_fd_for_current_cmd, STDIN_FILENO) which connects the looppipe stage and the after part.