"wget" Command Explanation

Wget command is used for downloading files from the web. It also provides us for downloading files from HTTP, HTTPS, FTP. Also, it can be seen as "Web Get". "Wget -v <URL>", gives information about the progress of download, URL information and HTTP response headers. Also, we can use this wget command to download files from web server without user logging on the system. Additionally, it can do all these things background of the current program.

Reason of choosing this command and flag

I picked this command (wget) because I am currently taking CS408 (Computer Networks) course and I wanted see the network system that is used through UNIX for download files. Also wget attracted me through downloading a files from my personal computer with using UNIX commands.

I picked '-v' because I wanted to see the information behind the HTTP requests are being sent to downloader. Additionally, I am interested in Backend Development and currently working on project and it is so much about requests. By this flag we can see the information about requests. Additionally, By this information we can catch some problems in the process of downloading particular files.

-A 4 => prints the 4 lines of current text => -v flag has 4 lines which also includes '- -verboose'

-w => I want to choose that only have '-v'. It needs to match with exact word in order to escape from words that includes '-v'. (Escaping from substring)

-e => We can search to specified pattern or REGEX in input. I used for searching specified input.

Process Hierarchy

At first my program gets pid from SHELL directly without using fork () explicitly but we also know that in main program we have pid (process id) and my program directly get pid without explicitly creating new child process. After printing SHELL process, I am initializing the pipe with fd[0] and fd[1]. (fd[1] : write, fd[0] : read). Then we are calling fork () in order to create new child process which will be used in MAN process. In the parent process of MAN (SHELL process), we are calling GREP process. As a result, MAN process and GREP process are sibling processes. We are calling MAN first, after that we are forking GREP process in SHELL process. We are duplicate fd[1] (writing side of pipe) to STDOUT (shell terminal output), by this approach we will write fd[1] which was going to be seen on shell terminal. After this duplication process, we will execute man command with execvp function. Now since we duplicate fd[1] (writing STDOUT but since we duplicate fd[1], in STDOUT file descriptor we will see fd[1] reference) we will see the result in the fd[1] instead of shell terminal. After that in parent process of MAN (SHELL process) we will fork() again as a process of GREP. Now we want to read so that we will duplicate fd[0] (read side of pipe) to STDIN (input side) then we will be able to use execution of grep command and we will write into output.txt. My man and grep processes can run concurrently because both of them is ran as a child of SHELL process and they are not waiting for each other. There is write "Hello" part in MAN process. This writing part will make GREP process to does not print "I'm GREP

process…" until it read this "Hello" from pipe. After reading this "Hello" from pipe in GREP process. Both MAN and GREP process will run concurrently because GREP will not be waiting for MAN to finish process because I am not using wait(NULL) or waitpid(…) functions. When MAN process starts, there will be read/write operations (read method blocking) occurs after this operations MAN and GREP will work concurrently since one is not waiting other one to finish the process (It will be exactly like in PA1 Process Hierarchy part but instead of Page 1 now we are waiting "Hello" read/write operations). SHELL process will wait MAN and GREP processes to finish their processes. After they finish their processes, SHELL process will print its pid and the program finishes.