Text

Description automatically generatedText

Description automatically generated

import os

import sys

def main():

cmds = [['cat', '/proc/cpuinfo'], ['echo', 'Hellow World'], ['python3', 'spinner.py', '1000000'], ['uname', '-a']]

for cmd in cmds:

rc = os.fork()

if rc<0:

print("Fork failed")

sys.exit(1)

elif rc == 0:

print("\nI am child. My pid ==", os.getpid(), ".\nExecuting", cmd)

os.execve('/usr/bin/'+cmd[0], cmd, os.environ)

else:

rc\_wait=os.wait()

print("I am parent. My pid ==", rc, ". Parent's pid ==", os.getpid())

last\_rc = os.fork()

if last\_rc > 0:

print("I am parent. My pid ==", last\_rc, ". Parent's pid ==", os.getpid())

exit()

else:

print("\nI am child. My pid ==", os.getpid(), ".\nExecuting python3 spinner.py 2000000")

os.execv('/usr/bin/python3', ['python3', 'spinner.py', '2000000'])

if \_\_name\_\_ == '\_\_main\_\_':

main()

cmds contains all the commands that need to be executed by the program and within the forloop, it will excute one by one. Since it requires one process(child process) for one command, fork is inside of forloop. If rc is less than 0, failed, if same as 0, successful child process, and greater than 0, the parent process is being running. If rc is 0, it will use execve in order to run the command. Meanwhile, the parent is in wait() since the child will run first. This steps will occurs 4 times for each commands. Afterall, as suggested in the assignment, child process will run python3 spinner.py 2000000’, however, the parent will not wait and the program will exit since there are no wait(). Interestingly, the child process yet processes and prints out after it is exited from the program since not deterministic when not using wait().