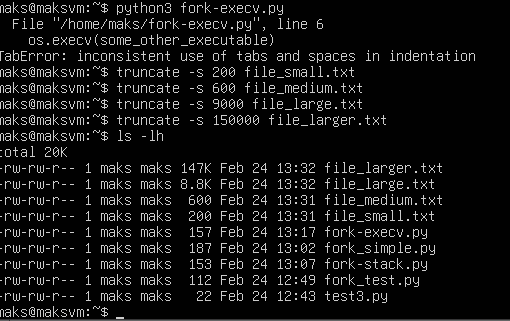
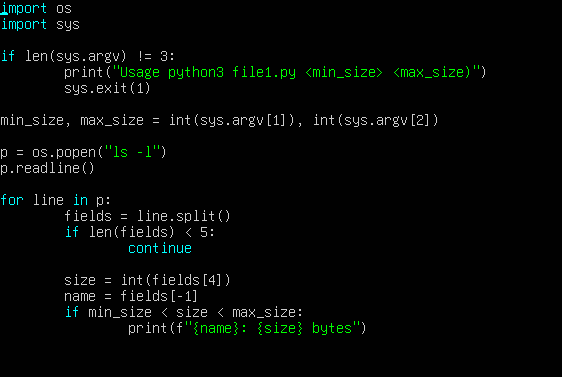
ECS518U Operating Systems

Lab 3: Introduction to Python Scripts and Creating Processes Using fork()

# PART A – Simple Python Scripts

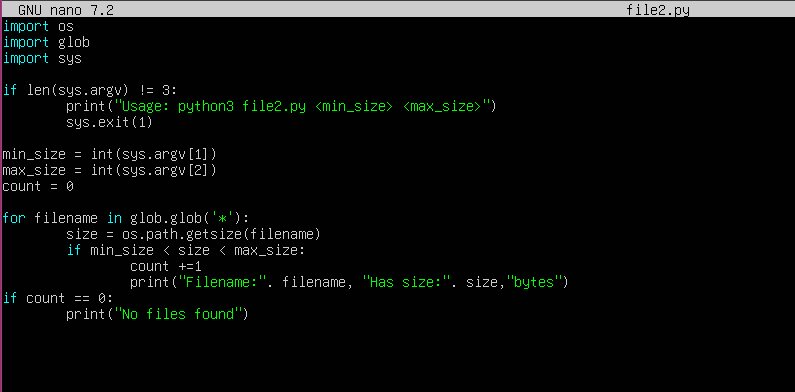


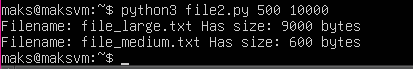
# Q1. Finding files with size greater than x bytes and less than y bytes: Method 1



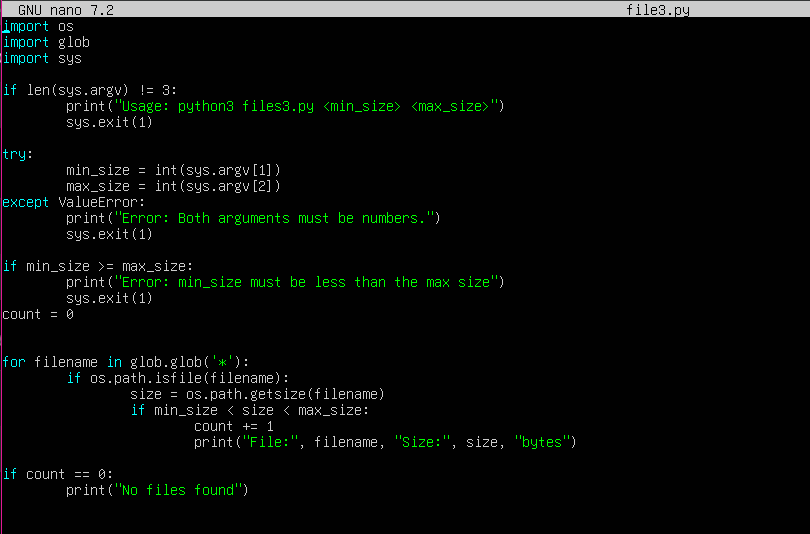
# 

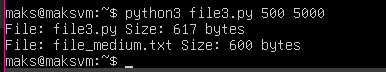
# Q2. Finding files with size greater than x bytes and less than y bytes: Method 2





# Q3. Processing command line arguments





# PART B – Process creation using fork

**Q1. Variables on the stack – what is the output? - (script fork-stack.py)**

This is based on an exam question from the 2018 May paper.

import os

temp = -1

new\_pid = os.fork()

if new\_pid == 0:

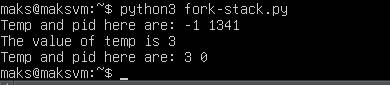
     temp = -3

print(“The value of temp is”, temp)

print(“Temp and pid here are:”, temp, new\_pid)

**Answer (fill in your answers below):**

* What will the output of this script be?



* Explain why the output is what it is. The explanation should be short.

Before forking the temporary value is set to -1. After the fork two separate processes, parent and child. Both contain a copy of temp and the new pid. In the child process temp is set to 3 and outputted. In the parent process temp is not changed.

**Q2. What is the output?**

There is no Python code given for this exercise. This is from the May 2022 exam paper.

Consider the simple program below.

Line 1 a = 0

Line 2 rv = os.fork()

Line 3 if(rv == 0) :

Line 4 print (a)

Line 5 a = -1

Line 6 os.execv(some\_other\_executable)

Line 7 print(“Execv is finished”)

Line 8 else :

Line 9 a = 1

Line 10 status = os.wait()

Line 11 print(a)

What would the output of this program be, assuming that calls to os.fork and os.execv succeed? Explain why the output will be as you say. **Write your answer below:**

The fork creates a child process, in the parent process rv is set to the PID of the child process which is zero. The child process then prints a as zero and then sets a to -1. Os.execv then executes another process which replaces the child process meaning it no longer executes any code. The parent process then sets a to 1 and waits for the child process to terminate before outputting the value 1. Therefore, The final output for the parent process is 1. The child process does not execute.