Usman Institute of Technology

Department of Computer Science Fall 2022

Name: Muhammad Waleed

Roll no: 20B-115-SE

Course: Operating Systems (CS312)

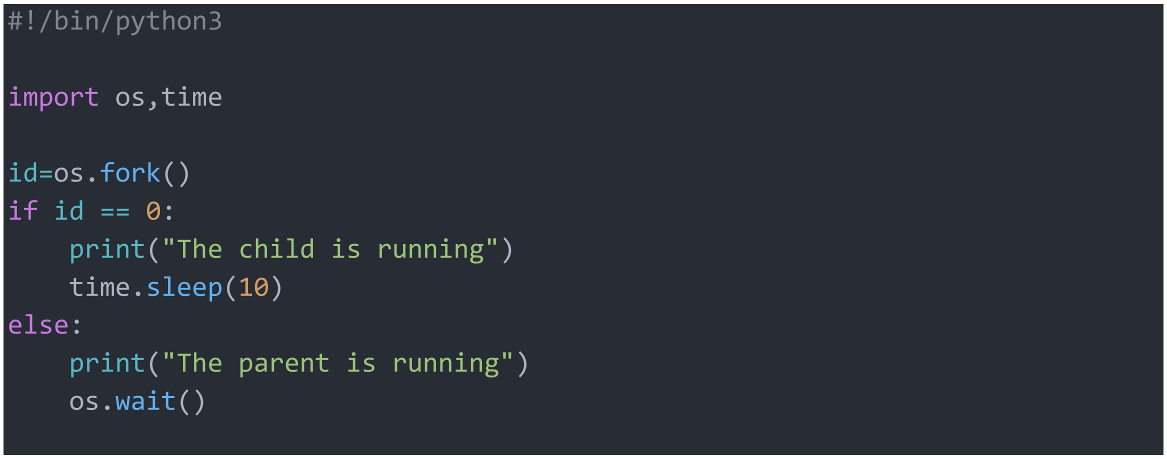
Course Instructor: Ma’am Shabina Mushtaq

Date: 17-Nov-2022

Muhammad Waleed
20b-115-se
Operating Systems
Lab#06

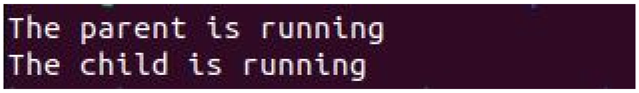
Lab Tasks:

1. Using a Linux system, write a program that forks a child process that ultimately becomes a zombie process. This zombie process must remain in the system for at least 10 seconds.



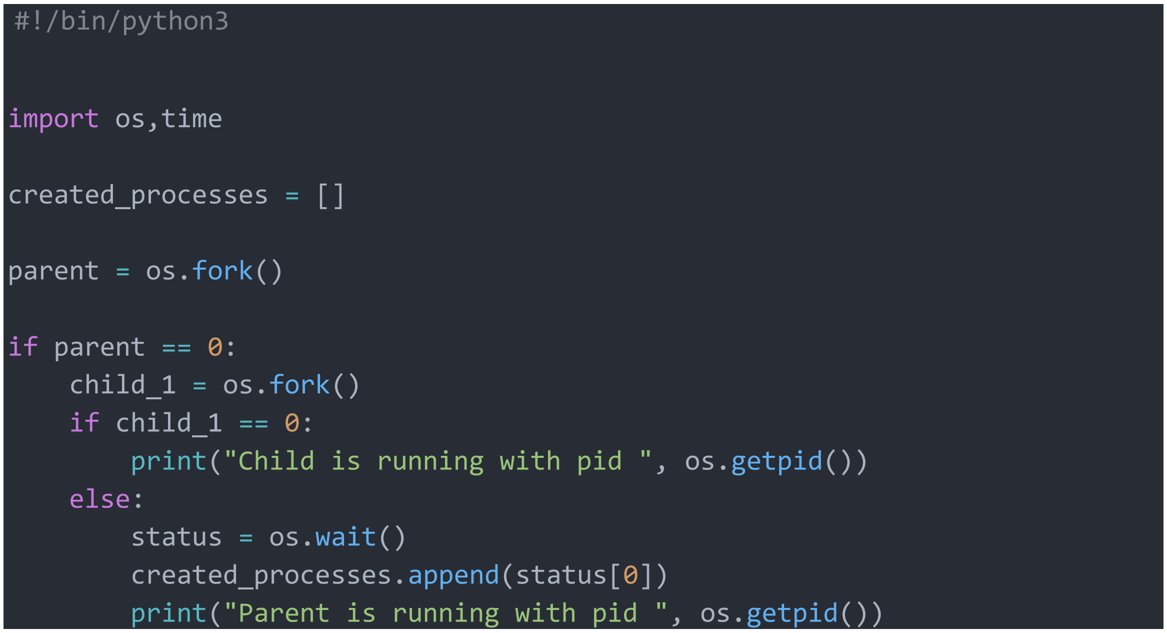
```
#!/bin/python3

import os,time

id=os.fork()
if id == 0:
    print("The child is running")
    time.sleep(10)
else:
    print("The parent is running")
    os.wait()
```

Output:

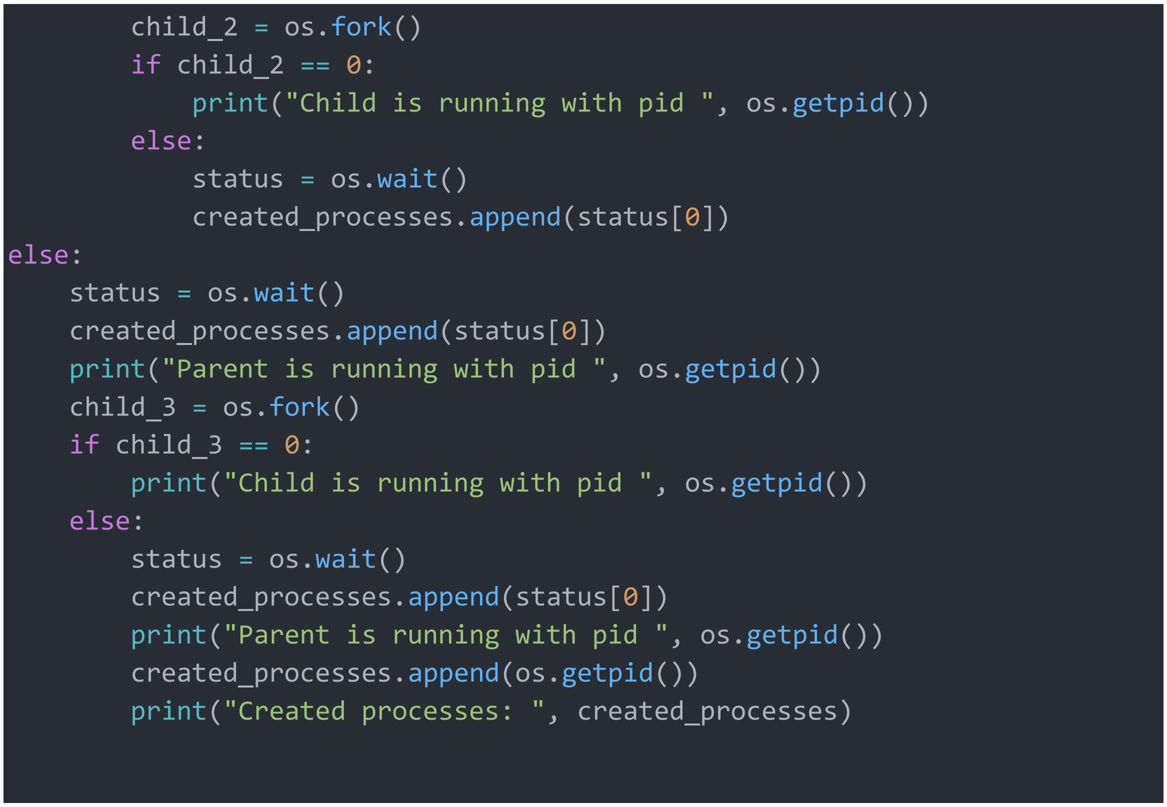


```
The parent is running
The child is running
```

2. Write a program that creates a child process which further creates its two child processes. Store the process id of each process in an array called Created Processes. Also display the process id of the terminated child to understand the hierarchy of termination of each child process.

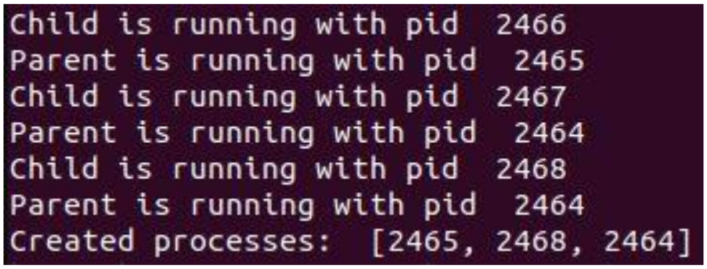


```
#!/bin/python3


import os,time

created_processes = []

parent = os.fork()

if parent == 0:
    child_1 = os.fork()
    if child_1 == 0:
        print("Child is running with pid ", os.getpid())
    else:
        status = os.wait()
        created_processes.append(status[0])
        print("Parent is running with pid ", os.getpid())
```



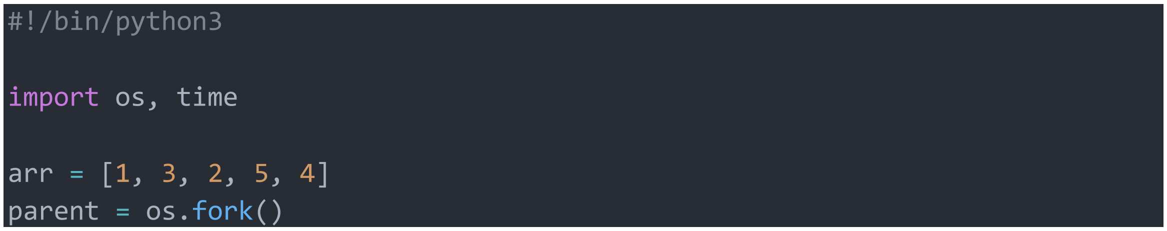
```
        child_2 = os.fork()
        if child_2 == 0:
            print("Child is running with pid ", os.getpid())
        else:
            status = os.wait()
            created_processes.append(status[0])
else:
    status = os.wait()
    created_processes.append(status[0])
    print("Parent is running with pid ", os.getpid())
    child_3 = os.fork()
    if child_3 == 0:
        print("Child is running with pid ", os.getpid())
    else:
        status = os.wait()
        created_processes.append(status[0])
        print("Parent is running with pid ", os.getpid())
        created_processes.append(os.getpid())
        print("Created processes: ", created_processes)
```

Output:



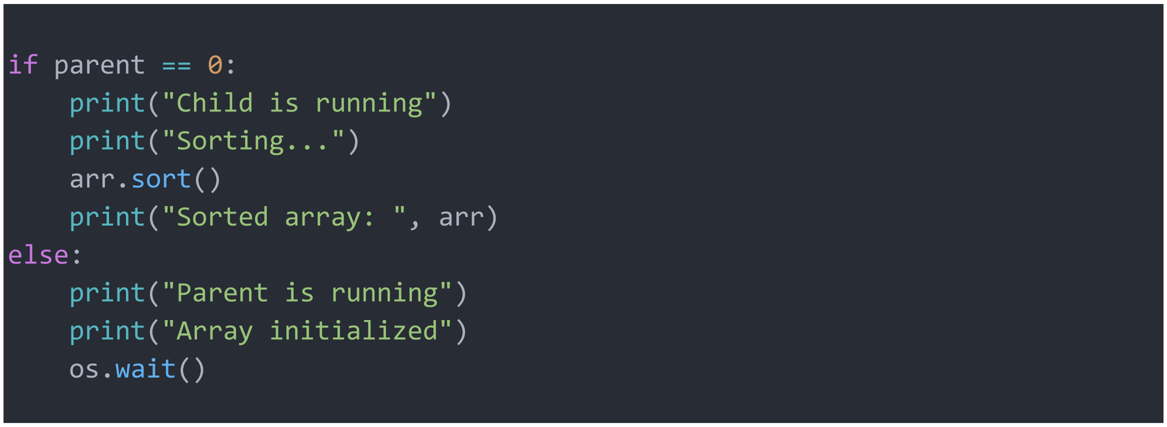
```
Child is running with pid  2466
Parent is running with pid  2465
Child is running with pid  2467
Parent is running with pid  2464
Child is running with pid  2468
Parent is running with pid  2464
Created processes:  [2465, 2468, 2464]
```

3. Write a program in which a parent process will initialize an array, and child process will sort this array. Use wait() and sleep() methods to achieve the synchronization such that parent process should run first.



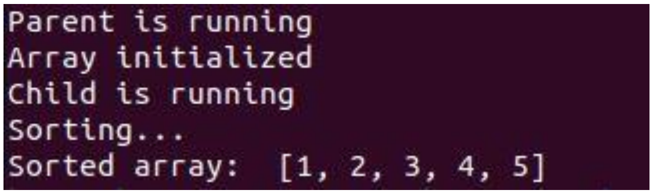
```
#!/bin/python3

import os, time

arr = [1, 3, 2, 5, 4]
parent = os.fork()
```

Muhammad Waleed
20b-115-se
Operating Systems
Lab#06



```
if parent == 0:
    print("Child is running")
    print("Sorting...")
    arr.sort()
    print("Sorted array: ", arr)
else:
    print("Parent is running")
    print("Array initialized")
    os.wait()
```

Output:



```
Parent is running
Array initialized
Child is running
Sorting...
Sorted array:  [1, 2, 3, 4, 5]
```