Nathan Mack
CSDS 338

Homework 1

A. LOOKUPS

1.

- IoT
  - Since 2020: 4,140
  - Since 2019: 9,760
  - Since 2016: 20,400
  - Since Anytime: 27,600

- Mobile
  - Since 2020: 8,580
  - Since 2019: 17,500
  - Since 2016: 43,000
  - Since Anytime: 391,000

2.

Technology	Since 2020	Since 2019	Since 2016	Since Anytime
IoT	4140	9760	20400	27600
Mobile	8580	17500	43000	391000
Security	9380	17900	45700	585000

3.

- IoT (w/ patents)
  - Since 2020: 6,420: 6,420 – 4,140 = 2,280
  - Since 2019: 15,500: 15,500 – 9,760 = 5,740
  - Since 2016: 24,300: 24,300 – 20,400 = 3,900
  - Since Anytime: 33,400: 33,400 – 27,600 = 5,800

- Mobile (w/ patents)
  - Since 2020: 21,700: 21,700 – 8,580 = 13,120
  - Since 2019: 18,500: 18,500 – 17,500 = 1,000
  - Since 2016: 126,000: 126,000 – 43,000 = 83,000
  - Since Anytime: 1,080,000: 1,080,000 – 391,000 = 689,000

4.
Patents

Technology	Since 2020	Since 2019	Since 2016	Since Anytime
IoT	2820	5740	3900	5800
Mobile	13120	1000	83000	689000
Security	12620	2300	48000	735000

5. There are significantly more search results for Silberschatz in comparison to the other two.

B. OPINIONS/THOUGHTS

1. I really appreciate the power that gives you in terms of low-level programming. As an electrical engineer with an interest in embedded software, C is much preferred alternative to something like assembly. One thing I dislike is that C is still not the easiest code to write in comparison to a language like Python.

2. When a parent process dies before it’s children, the kernel puts them under the care of init. This action occurs because the kernel knows there will be no parent give a wait call. Init performs the wait system call so the child processes can die.

3. The V command gives use a forest view of the currently running processes on the system.

4. A few of the numbers are consistently increasing, and once or twice every minute a shift is seen in all of the numbers. I believe this is due to different process running and terminating.

5. I chose the kill signal SIGCONT, my friend chose SIGQUIT

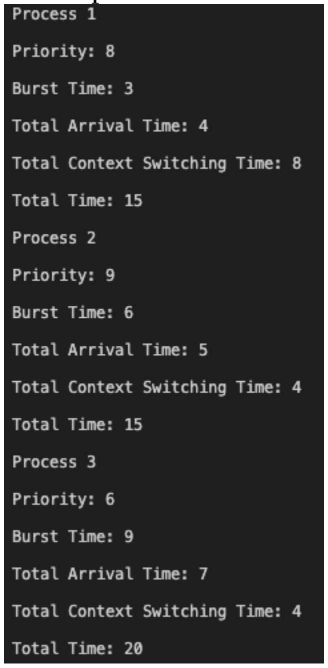
SIGQUIT is bigger and badder because it actually kills an entire process, while SIGCONT just continues a process.

C. SIMULATIONS

a.) Refer to simulation_a.py

b.) Refer to simulation_b.py

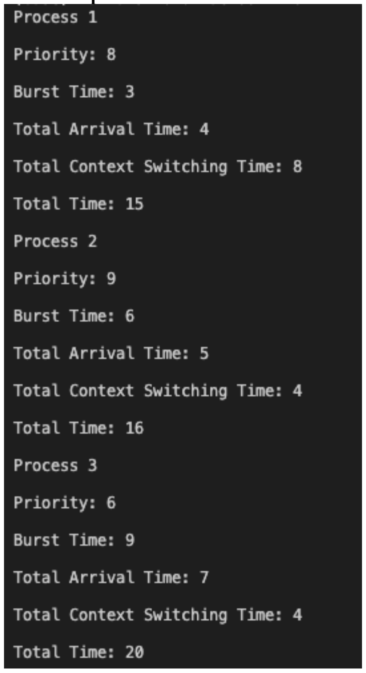
c.)
A. Output



```
Process 1
Priority: 8
Burst Time: 3
Total Arrival Time: 4
Total Context Switching Time: 8
Total Time: 15
Process 2
Priority: 9
Burst Time: 6
Total Arrival Time: 5
Total Context Switching Time: 4
Total Time: 15
Process 3
Priority: 6
Burst Time: 9
Total Arrival Time: 7
Total Context Switching Time: 4
Total Time: 20
```

A. Average Response Time = (15 + (15 – 4) + (20 – 5)) / 3 = 13.67 seconds

B. Output



```
Process 1
Priority: 8
Burst Time: 3
Total Arrival Time: 4
Total Context Switching Time: 8
Total Time: 15
Process 2
Priority: 9
Burst Time: 6
Total Arrival Time: 5
Total Context Switching Time: 4
Total Time: 16
Process 3
Priority: 6
Burst Time: 9
Total Arrival Time: 7
Total Context Switching Time: 4
Total Time: 20
```

B. Average Response Time = (15 + (16 – 4) + (20 – 5)) / 3 = 14.00 seconds

d.)

A. Average Burst Completion Time = ((15 – 4) + (15 – 6) + (20 – 7)) / 3 = 11.00 seconds

B. Average Burst Completion Time = ((15 – 4) + (16 – 6) + (20 – 17)) / 3 = 11.33 seconds

e.)

A. % Time Context Switching: (8 + 4 + 4) / (34) = 47%

B. % Time Context Switching: (8 + 4 + 4) / (35) = 46%

D. C CODING

1. Program and output are below

```
#include <stdio.h>

int main()
{
   char name[] = "Nathan";
   int nameSize = sizeof(name) / sizeof(name[0]);

   for (int i = 0; i < nameSize; i++)
   {
      printf("%c\n", name[i]);
   }
}
```

```
N
a
t
h
a
n
```

2. 32512 pts/19   00:00:00 sleep => The processes from ps command

```
#include <stdlib.h>
#include <unistd.h>
#include <stdio.h>

int main()
{
  sleep(100);
}
```

3.

```
#include <stdlib.h>

int main()
{
   for (int i = 0; i < 10; i++)
   {
      system("sleep 100 &");
   }
}
```

```
6606 pts/19   00:00:00 sleep
6608 pts/19   00:00:00 sleep
6610 pts/19   00:00:00 sleep
6612 pts/19   00:00:00 sleep
6614 pts/19   00:00:00 sleep
6616 pts/19   00:00:00 sleep
6618 pts/19   00:00:00 sleep
6620 pts/19   00:00:00 sleep
6622 pts/19   00:00:00 sleep
6624 pts/19   00:00:00 sleep
```

You have fg control immediately after running the program. All of the processes take 100 seconds to execute.

4.

```
#include <stdlib.h>

int main()
{
   for (int i = 0; i < 10; i++)
   {
      system("sleep 100");
   }
}
```

It takes 1000 seconds until ps can be used and the sleep processes completed.

5.

```
#include <stdlib.h>

int main()
{
   for (int i = 0; i < 1000; i++)
   {
      system("echo A >> out.a.txt");
   }
}
```

0.044, 0.052, 0.056, 0.052, 0.048, 0.048, 0.052, 0.052, 0.052, 0.052

6.

Average of above numbers = 0.051

1000 / 0.051= 19608

The number of iterations ends up being around 17,000. This becomes difficult because the ideal number of 10 seconds is harder to reach as we increase the file size.

7.

User and sys times don't add up to the real time because they are in terms of the number of seconds that the CPU uses rather than execution time.

8.

The time command has a large number of resource specifiers that can give times of other things going on in an operating system, such as number of socket messages received by a process.

9.

The date command has specific formatting codes that controls the output.

10.

It might be wise to delete the out.a.txt file after every test so that it contains a standard amount of memory for each test. Additionally, this makes the file take up more space.