AO Coursework

by Michael Mbomena

Submission date: 04-Mar-2022 06:03AM (UTC+0000)
Submission ID: 172610036
File name: AO_Coursework_March2022.pdf (427.94K)
Word count: 1414
Character count: 6966

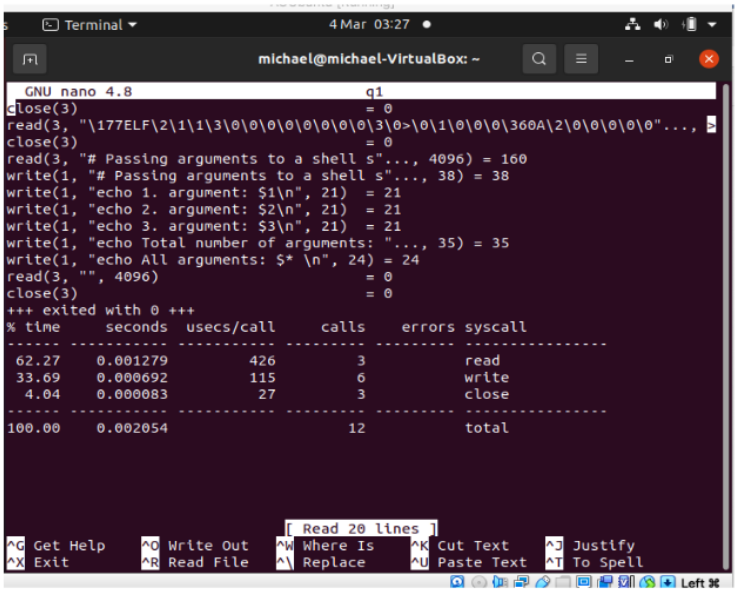
Module Code: CS2AO17
Assignment report Title: Operating System Lab practical
Student Number: 29017995
Date (when the work completed): 3 March 2022
Actual hrs spent for the assignment: 15

Preliminaries

Investigating and experimenting concepts such as system calls, CPU processes and file systems. Using C programming language in a Linux environment. This report is being written to document my findings and learnings.

Q1. Strace output & summary

Command: $ strace -o q1 -C -e trace=open,close,read,write ./mycat cmd2.sh



Strace runs the specified command until it exits and records the system calls. Each line contains the system call name. The arguments are in parenthesis and its return values are at the end. We can see what system calls The command '-e trace=open,close,read,write' is used to trace multiple system calls and these are the only ones we can see. The '-C' command is used to generate a report showing total time, calls and errors for each system call. The '-o' command places the output in the q1 file. We can see that the system call read takes the majority of the time. The output shows us that the parameter read 160 bytes of data. The system call 'write' is called the most. Close() free the file descriptor, as the file is mapped successfully into process virtual memory and no more accessing through a file descriptor is required. At the end of the data we see that the return values of read and close are 0 which means that they are a success. We can see the total number of calls and how long it took.

Q2. Bash command explained

Bash is a shell or a command-line interpreter. Bash command allows you to execute commands and script files that can process things and do certain functions that include file management, starting programs, executing its own code.

Q3. Forking

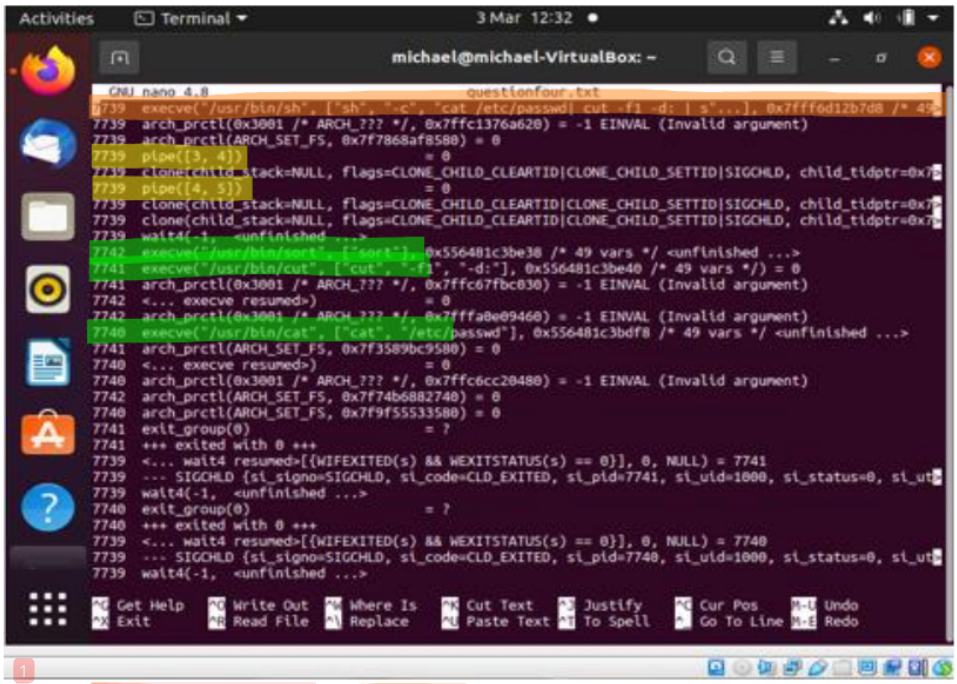
Q3.c

```
#include <stdlib.h>
#include <unistd.h>
#include <stdio.h>
#include <sys/wait.h>

int main(int argc, char *argv[])
{
	int counter = 0;
	int child1 = fork();

	// if child == 0, then it is the child process
	if (child1 == 0)
	{
		for (counter = 0; counter < 10; counter++)
		{
			printf("child %d process, counter = %d\n", getpid(),
counter);
			sleep(1);
		}
	}
	else // this is the parent process
	{
		// create a second child
		int child2 = fork();

		// if child2 == 0, the current process is the second child
		if (child2 == 0)
		{
			for (counter = 0; counter < 10; counter++)
			{
				printf("child %d process, counter = %d\n",
getpid(), counter);
				sleep(1);
			}
		}
		else // parent process
		{
			for (counter = 0; counter < 10; counter++)
			{
				printf("parent %d process, counter = %d\n",
getppid(), counter);
				sleep(1);
			}
		}
	}
}
```

The code output alternates because it is a loop and always goes back to the parent then to the two children.

2

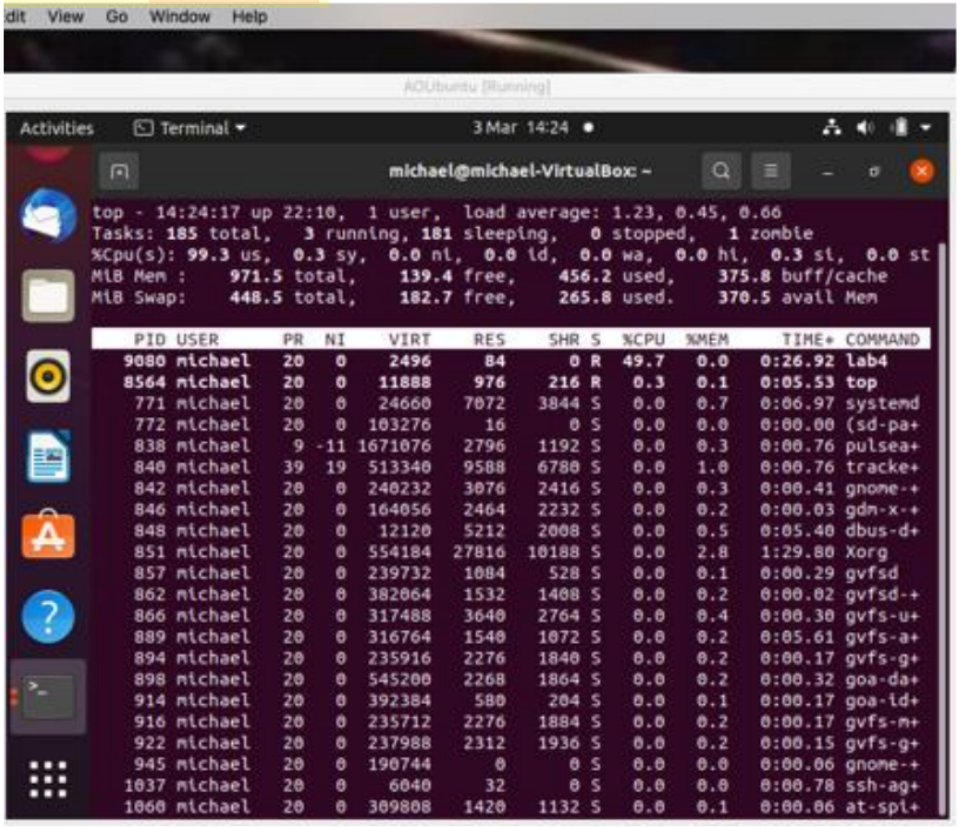
Q4. Strace Output



1

i) New process creation is highlighted
ii) Execution is highlighted
iii) Pipes are setup

Q5. CPU Processes



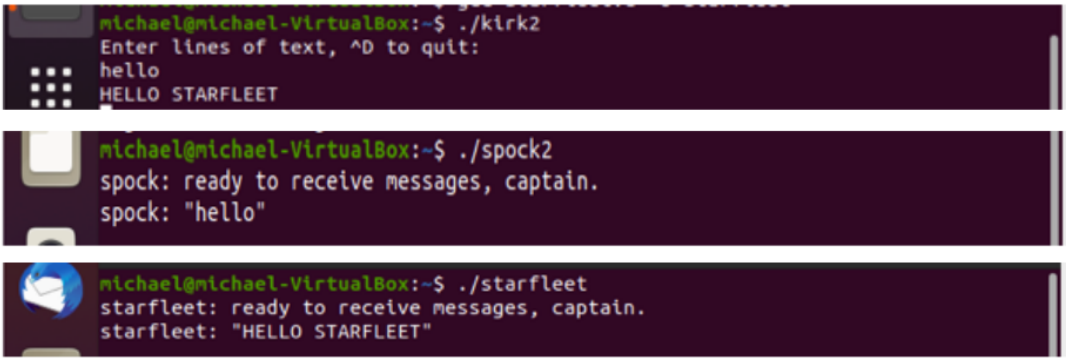
Utilisation Table

N	CPU% Used
1	99.7
2	49.5
4	24.2
6	16.6
8	12.3
16	6.3

With regards to the CPU core when you increase processes you end up with less CPU performance. If both processes are at 100% that would mean that the CPU has two cores. If

you have 4 four processes, then the CPU% would decrease. I found out my computer has 1 core as when only 1 process is started 98% of the CPU is used. You can deduct from measurements how much processing resources the computer has by seeing how much of the CPU% is used by the processes. If one PC runs two processes all at 100% and one runs two processes at 50%, then the first PC has double to processing power. If you can conduct many processes while only using a small amount of the CPU that means the computer has a good amount of processing power.

Q6. Messages



Kirk2.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include <string.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>

struct my_msgbuf {
    long mtype;
    char mtext[200];
};

int main(void)
{
    struct my_msgbuf buf;
    int msqid_spock;
    key_t key_spock;
    int msqid_starfleet;
    key_t key_starfleet;

    if ((key_spock = ftok("kirk2.c", '1')) == -1) { /*this connects to spock*/
        perror("ftok");
        exit(1);
}

    if ((msqid_spock = msgget(key_spock, 0644 | IPC_CREAT)) == -1) {
        perror("msgget");
        exit(1);
```

```
    }

    if ((key_starfleet = ftok("kirk2.c", '2')) == -1){ /*this connects to Starfleet*/
        perror("msgget");
        exit(1);
    }

    if ((msqid_starfleet = msgget(key_starfleet, 0644 | IPC_CREAT)) == -1){
        perror("msgget");
        exit(1);
    }



    printf("Enter lines of text, ^D to quit:\n");

buf.mtype = 1; /* we don't really care in this case */

    while(fgets(buf.mtext, sizeof buf.mtext, stdin) != NULL) {
        int len = strlen(buf.mtext);

        /* ditch newline at end, if it exists */
        if (buf.mtext[len-1] == '\n') buf.mtext[len-1] = '\0';



    int caps = 1; /* checks message is all caps*/
    for (int i = 0; i < len - 1; i++)
    { if (buf.mtext[i] > 96)
        {
           caps = 0;
        }
}

if (caps)
{if (msgsnd(msqid_starfleet, &buf, len + 1, 0) == -1)
        perror("msgsnd");
}
else /* send to spock*/
{
    if (msgsnd(msqid_spock, &buf, len + 1, 0) == -1)
        perror("msgsnd");
}
}

    if (msgctl(msqid_spock, IPC_RMID, NULL) == -1) {
        perror("msgctl");
        exit(1);
    }

    return 0;
}
```

Spock2.c

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

```c
#include <errno.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>

struct my_msgbuf {
    long mtype;
    char mtext[200];
};

int main(void)
{
    struct my_msgbuf buf;
    int msqid;
    key_t key;

    if ((key = ftok("kirk2.c", '1')) == -1)
{
        perror("ftok");
        exit(1);
    }

    if ((msqid = msgget(key, 0644)) == -1)
{
        perror("msgget");
}

    printf("spock: ready to receive messages, captain.\n");

    for(;;) { /* Spock never quits! */
        if (msgrcv(msqid, &buf, sizeof buf.mtext, 0, 0) == -1) {
            perror("msgrcv");
            exit(1);
        }
        printf("spock: \"%s\"\n", buf.mtext);
    }

    return 0;
}
```

Starfleet.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>

struct my_msgbuf {
    long mtype;
    char mtext[200];
};

int main(void)
{
    struct my_msgbuf buf;
    int msqid;
    key_t key;

    if ((key = ftok("kirk2.c", '2')) == -1) {  /* same key as kirk2.c */
        perror("ftok");
        exit(1);
    }
```

```
    if ((msqid = msgget(key, 0644)) == -1)
        perror("msgget");
}

    printf("starfleet: ready to receive messages, captain.\n");

    for(;;) { /* Starfleet never quits! */
        if (msgrcv(msqid, &buf, sizeof buf.mtext, 0, 0) == -1) {
            perror("msgrcv");
            exit(1);
        }
        printf("starfleet: \"%s\"\n", buf.mtext);
    }

    return 0;
}
```

Instead of one key just for Spock, there are two keys made for Spock and Starfleet. They set so the keys are able to connect to Kirk2. They are both given different identities. Identities '1' & '2' respectively. If spock and Starfleet can connect then they are able to receive messges using 'msgget'. Then caps is intialised to equal 1 to check that message is all caps. It starts at the first letter and goes through every letter one by one throughout the string to check if any character is greater than 96, as 97 is when lower case letters start on the ascii table. If any characters are greater than 96 then caps = 0 and the message is sent to Spock. If no haracters are greater than 96 then the message is sent to Starfleet.

To make Starfleet code from Spock is used. Msgget() is used to connect to the queue as it returns the message queue ID on success. The ftok() function generates a key from two arguments for msgget(). To connect they have to have the same key. To receive the function msgrcv() needs to be used. As the msgtype is 0 it means that it receives the next message on the queue, regardless of its type which is what we want. When the message is reived it its printed out.

Q7. Memory Management

Vaddr2.c

```
#include <stdlib.h>
#include <stdio.h>
int main(int argc, char *argv[])
{
    unsigned long page, offset, address;
    if (argc != 2) exit(1);
    address = atoll(argv[1]);

    page = address >> 20;       /*calculates page number 2^20 = 1 million*/
    offset = address & 0xfffff; /*calculates remaining offset 0xfffff can holds many
values*/
    printf("The address %lu contains: \n", address);
    printf("page number = %lu\n", page);
    printf("offset = %lu\n", offset);
    return 0;
}
```

Address is the address that the file should be mapped into. The offset is the the offset in the file that it should be mapped from. 1MB is 1 million bytes and 2^20 is 1 million.

Summary

References

https://www.tutorialspoint.com/unix_commands/bash.htm
https://man7.org/linux/man-pages/man1/strace.1.html
https://stackoverflow.com/questions/55873277/creating-n-child-process-in-c-using-fork

AO Coursework

ORIGINALITY REPORT

40%	31%	9%	35%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to University of Reading Student Paper	28%
2	www.gadgety.net Internet Source	5%
3	Igor Ljubuncic. "Getting geeky – tracing and debugging applications", Elsevier BV, 2015 Publication	2%
4	www.geeksforgeeks.org Internet Source	2%
5	forensics.sans.org Internet Source	1%
6	jorge.fbarr.net Internet Source	1%
7	beej.us Internet Source	1%
8	www.ciciliani.com.ar Internet Source	1%

Exclude quotes	On	Exclude matches	Off
Exclude bibliography	On		

AO Coursework

GRADEMARK REPORT

FINAL GRADE

78/100

GENERAL COMMENTS

Instructor

PAGE 1

PAGE 2

Comment 1

good explanation but citation is needed when using external source of information

PAGE 3

Comment 2

good job highlighting the requirements but the command itself is missing.

PAGE 4

Comment 3

It is not clear from the screenshot how many processes have been run and the size of *N*

PAGE 5

PAGE 6

Comment 4

This line is not clear, why 96 ?

PAGE 7

PAGE 8

Comment 5

Empty, you better remove it.

RUBRIC: CS2AO17 - CW-OS 21-22_AMENDED 78 / 100

Q1		8 / 8
NOVICE (0)	No Credit	
COMPETENT (3)	Incomplete or erroneous	
PROFICIENT (6)	Correct □□□□□□ output summary provided but no/wrong description	
EXCEPTIONAL (8)	- Correct □□□□□□ output summary provided - Description of the output is given	

Q2		5 / 7
NOVICE (0)	No Credit	
COMPETENT (3)	Incomplete or erroneous explanation of bash command	
PROFICIENT (5)	Partial explanation	
EXCEPTIONAL (7)	□□□□ command explained	

Q3		10 / 14
NOVICE (0)	No Credit	
COMPETENT (6)	Incomplete code, code description or output explanation	
PROFICIENT (10)	- Program code is provided - Partial/insufficient comments/description of the code is given - Partial/insufficient Description of the output	
EXCEPTIONAL (14)	- Program code is provided - Comments/description of the code is given - Description why code output alternates	

Q4		6 / 10
NOVICE (0)	No Credit	
COMPETENT (4)	Incomplete or erroneous output and/or explanation	

PROFICIENT (6)	- Complete strace output is provided with partial explanation of process creation or execution or pipes
EXCEPTIONAL (10)	- Complete strace output is provided - New process creation is highlighted - Execution is highlighted - Pipes are setup

Q5 12 / 16

NOVICE (0)	No Credit
COMPETENT (6)	- processor utilisation table is wrong or not provided. - Erroneous reasons for 100% CPU utilization. - Erroneous deduction of available processing resources
PROFICIENT (12)	- Provided processor utilisation table for N=2,4,8,16 - one of the following is missing or erroneous: - Reasons for ~100% CPU utilization - Deduction of available processing resources
EXCEPTIONAL (16)	- Provided processor utilisation table for N=2,4,8,16 - Reasons for ~100% CPU utilization - Deduction of available processing resources

Q6 12 / 16

NOVICE (0)	No Credit
COMPETENT (6)	ncomplete or erroneous code for kirk2.c or spock2.c and startfleet.c or wrong/incomplete description the code and the msg recipient decision
PROFICIENT (12)	- Urgent code for kirk2.c is provided. - on of the following is missing: - Description of the code given - Code for spock2.c and starfleet.c is provided - Description of the code given, explaining msg recipient decision
EXCEPTIONAL (16)	- Urgent: Code for kirk2.c is provided - Description of the code given - Code for spock2.c and starfleet.c is provided - Description of the code given, explaining msg recipient decision

Q7 9 / 9

NOVICE (0)	No Credit
COMPETENT (0)	Erroneous or incomplete
PROFICIENT (6)	- Incomplete or missing vaddr2.c - Wrong description while the code is correct.
EXCEPTIONAL (9)	- Code for vaddr2.c is provided - Description of the code is given

NOVICE (6)	Unsatisfactory e.g., blurry artifacts, no proper formatting, grammatical mistakes, lengths not appropriate
COMPETENT (12)	minor issues but also some good features
PROFICIENT (16)	Well organized, good artifacts, good text, nice formatting
EXCEPTIONAL (20)	Exceptional (almost) perfect, no potential for improvement