UPPSALA
UNIVERSITET

Report for 1DT086 and 1DT032

Lab 3: PROCESSES AND MULTICORES

Group 11

Lingyan Duan Anwar Chowdhury
Nithesh Chandher Karthikeyan

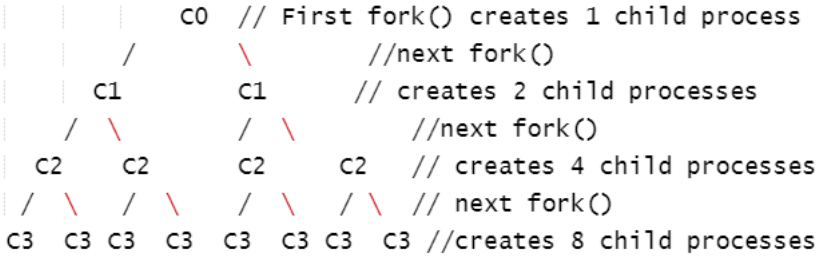
October 4, 2019

1 Creating some processes

Question 1.1. How many new processes will a program that contains the code fork(); fork(); fork(); fork(); create? Explain your answer clearly.

Answer to 1.1 : It creates 15 new processes when we run the code fork(); fork(); fork(); fork();

No of children = 2 power of n where n= number of time fork() gets called.



Question 1.2. What is the difference between the code snippets in the below listings with respect to the parent-child relationship of the created processes?

Answer to 1.2 : In the first snippet, parent process runs the else statement and the pid becomes 0, so the child process runs the if statement. Since there is one more fork() in the if statement, pid becomes 0 again and the child process runs the if statement again.

In the second snippet, parent process runs the else statement first and the fork() in else statement gets called. So pid becomes 0 and child process runs the if statement. After running the if statement pid is not equal to zero. So it goes back and runs the else statement.

Listing 1: C program that maps the LED matrix framebuer device into its memory space

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <stdint.h>
4  #include <time.h>
5  #include <sys/types.h>
6  #include <sys/wait.h>
7  #include <unistd.h>
8  #include "led_matrix.h"
9  int main()
10 {
11 open_led_matrix();
```

```
12 int pid = fork();
13 if (pid == 0)
14 {
15 printf("I'm the child! Lighting LED at (1,2)...\n");
16 set_led(1,2, RGB565_GREEN);
17 }
18 // parent process because return value non-zero.
19
20 else
21 {
22 printf("I'm the parent! Lighting LED at (3,2)...\n");
23 set_led(2, 1, RGB565_RED);
24 wait(NULL);
25 usleep(2000000);
26 clear_leds();
27 close_led_matrix();
28 }
29 return 0;
30 }
```

2 Processes and scheduling

Question 2.1. Describe the progress of the child processes that you can observe on the LED matrix, for the different number of processes that run concurrently. Explain clearly why they behave in this way considering the default scheduling and the number of cores in the Raspberry Pi's processor.

Answer to 2.1 : When we execute the program, it light up first rows LED's of led display randomly .After that 1st rows will be remain turned on and then again second row led turned on randomly. When all the lights on in 2nd row, again 3rd row turned on randomly. Rest of the rows will repeat same thing. The randomness is because we are not scheduling child processes.

Question 2.2. Describe the progress of the child processes that you can now observe on the LED matrix, for the different number of processes that run concurrently. Explain clearly why they behave in this way considering the nice-values and the number of cores in the Raspberry Pi's processor.

Answer to 2.2 : When we executed our program , the LED's in row 1 turns on one by one sequentially .After all the LED's in row one lights up, again the led of 2nd row turned on sequentially one by one. The same thing will be done up to the last row .Since we have included nice function which schedules the child processes and gets sequentially lights up.

Listing 2: C program that creates processes that perform some computations to observe how they progress and how they are scheduled.

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

```
2  #include <stdlib.h>
3  #include <stdint.h>
4  #include <time.h>
5  #include <sys/types.h>
6  #include <sys/wait.h>
7  #include <unistd.h>
8  #include "led_matrix.h"
9
10 int col;
11 /*
12 * Do some pointless CPU - heavy computations . Takes about 1 second
13 * to complete on a single core of the ARM Cortex A53 processor of
14 * the Raspberry Pi 3 model B with default gcc 6.3.0 op tim iz at io ns .
15 */
16 void pointless_calculation () {
17 int amount_of_pointlessness = 100000000;
18 int x = 0;
19 for ( int i = 0; i < amount_of_pointlessness ; i ++) {
20 x += i;
21  }
22 }
23 void run_child(int n) {
24 for (col = 0;col<=8;col++) {
25  open_led_matrix();
26  pointless_calculation ();
27  usleep (1000000) ;
28  set_led(n, col, RGB565_RED);
29 }
30 clear_leds();
31 close_led_matrix();
32 }
33
34 int main() {
35     int n,num_children,j;
36     open_led_matrix();
37     for(num_children = 0;num_children<=8;num_children++){
38     for(n = 0;n<num_children;n++)
39     {
40         int pid=fork();
41         if(pid==0){
42         nice(n);
43         run_child(n);
44         exit(0);
45         }
46     }
47     for(j=0;j<=num_children;j++)
48     {
49     wait(NULL);
50     }
51     usleep(1000000);
52     clear_leds();
53
54 }
55    close_led_matrix();
56    // run_child(n);
57    return 0;
58 }
```

3 Write a simple shell

Listing 3: A simple shell (like, for example, bash) that can be used to start other programs

```
1
2  #include <stdio.h>
3  #include <stdlib.h>
```

```c
#include <string.h>
#include <stdint.h>
#include <time.h>
#include <unistd.h>
#include <sys/wait.h>


int main()
{
char command[100];
//int i = 0;
do
{
printf("myshell > ");
scanf("%s",command);

//command == ls
if(strcmp(command, "ls") == 0)
{
    int pid = fork();
    if(pid == 0)
    {
        execl("/bin/ls", "ls", (char *) NULL);

    }
    else{
    wait(NULL);}

}else if(strcmp(command, "pstree")== 0)
{
    int pid = fork();
    if(pid == 0)
    {
        execl("/usr/bin/pstree", "pstree", (char *) NULL);
    }
     else{
        wait(NULL);}

}else if(strcmp (command, "htop")== 0)
{
    int pid = fork();
    if(pid == 0)
    {
        execl("/usr/bin/htop", "htop", (char *) NULL);
    }
     else{
        wait(NULL);}

}else if(strcmp(command, "ifconfig")==0)
{
    int pid = fork();
    if(pid == 0)
    {
        execl("/sbin/ifconfig", "ifconfig", (char *) NULL);
    }
     else{
        wait(NULL);}

}else if(strcmp(command,"quit")==0)
{
    printf("quitting...\n");
}
else
{
    printf("Command Unknown\n");
}
}while(strcmp(command,"quit")!=0);

}
```

UPPSALA
UNIVERSITET