PL1 - Processes and Exec functions

Luís Nogueira, Orlando Sousa

{lmn, oms}@isep.ipp.pt

Sistemas de Computadores
2024/2025

1. Consider the following code:

```c
#include<stdio.h>
#include<sys/types.h>
#include<unistd.h>

int main(void){
  pid_t p, a;
  p = fork();
  a = fork();
  printf("Concurrent Programming\n");
  return 0;
}
```

   a) How many processes are created? Draw a process tree to justify your answer.

   b) How many times is the phrase "Concurrent Programming" displayed on the screen? Provide a justification for your answer.

   c) Change the program to handle all possible error situations correctly.

2. Create a program that spawns two child processes and performs the following tasks:

   - In the parent process, print "I am the father" along with its process ID (PID). The father should wait for the first child to finish and then wait for the second child. Additionally, ensure that the program checks if the children terminated normally and present the exit value of each child process.
   - In the first child process, print "I am the first child" along with its PID. This process should sleep for 5 seconds and then return an exit value of 1.

- In the second child process, print "I am the second child" along with its PID. This process should return an exit value of 2.

Open two shell windows: one to execute the program and another to display the process table. Run your program in one window and, in the other window, execute the `ps` command several times to check the process table (see `man ps` for more information). Review the values, paying particular attention to the new table entries and the status of each created process throughout its lifecycle.

3. Consider the following code:

```
#include<stdio.h>
#include<sys/types.h>
#include<unistd.h>

int main(void){
  pid_t  pid;
  int  i;

  for(i=0;i<4;i++)
    pid = fork();

  printf("Concurrent Programming\n");
  return  0;
}
```

   a) How many processes are created? Draw a process tree to justify your answer.

   b) How many times is "Concurrent Programming" displayed on the screen? Provide a justification for your answer.

4. Consider the following code:

```
#include<stdio.h>
#include<unistd.h>
#include<sys/types.h>

int main(void){
  pid_t  pid;
  int  f;

  for(f=0;f<3;f++){
    pid = fork();
    if (pid > 0){
      printf("I am the father\n");
    }
    else{
      sleep(1);
    }
```

```
  }

  return  0;
}
```

a) Review the code and determine how many processes will be the created.

b) Test the program to verify your answer to the previous question. Run the `ps` command several times to observe status changes in the created processes.

c) Draw the corresponding process tree.

d) Modify the program to create only 3 children.

e) Modify the program to handle all possible error conditions.

f) Modify the program so that the parent process waits for each of its children to finish.

g) Modify the program so that each child process returns its order number to the parent process. The parent should print the value returned by each child process, along with its pid.

h) Modify the program so that the parent waits for the second child without blocking.

5. Consider the following code:

```
#include<unistd.h>
#include<stdio.h>
#include<sys/types.h>

int main(void){
  fork();
  printf("1\n");
  fork();
  printf("2\n");
  fork();
  printf("3\n");
  return  0;
}
```

a) How many processes are created? Draw a process tree to justify your answer.

b) Is it possible for the number 1 to appear on the screen after a 3? If so, why?

6. Create and initialise an array `int data[100 000]` with random values between 1 and 10, and spawn a new process:
   - The two processes (parent and child) perform the following calculation: `result[i] = data[i]*4 + 20;`
   - Each process handles the calculation for 50,000 positions, storing the result in a `result` array;
   - The results should be presented sequentially based on the array index (0-49999 and then 50000-99999).

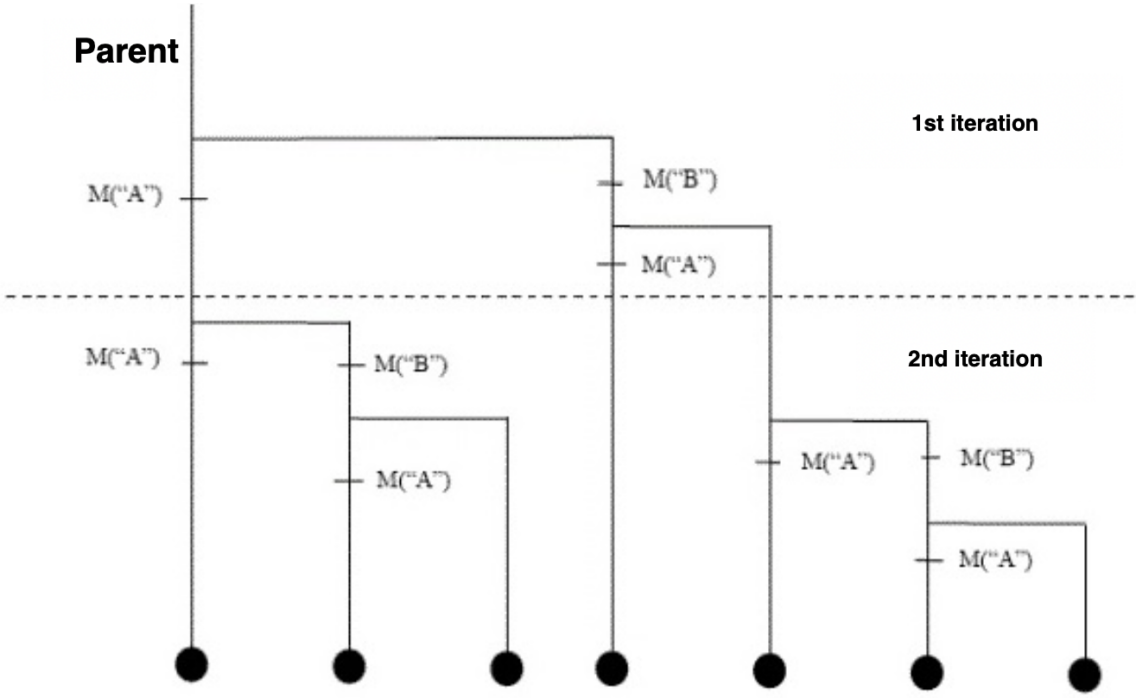
   Note: the calculations must be carried out in a concurrent/parallel manner. Only the presentation of the results must be sequential.

   a) Based on you current knowledge, can the `result` array be shared by the parent and child processes? Justify your answer.

   b) Provide a solution to the problem.

   c) Execute the program, and in another shell window, use the `ps` command to monitor the progress. Provide commentary on the results.

7. Implement a program to replicate the progression depicted in the following diagram:



Note: The function `void M (char *)` displays a string on the screen.

8. Carefully analyse the following program.

```
#include<stdio.h>
#include<unistd.h>
#include<sys/types.h>

int main(void){
  int a,b,c,d;

  a=0;
  b=fork();
  c=getpid();
  d=getppid();
  a=a+5;
  printf("\na=%d, b=%d, c=%d, d=%d\n",a,b,c,d);
}
```

   a) Identify the variables (a, b, c, and d) that will have identical values in both processes.

   b) Support your answer by drawing a process tree.

9. Implement a program that creates 6 child processes.

   - Each child process writes 200000 numbers in the screen:
     - Child 1: 1 to 200000
     - Child 2: 200001 to 400000
     - Child 3: 400001 to 600000
     - Child 4: 600001 to 800000
     - Child 5: 800001 to 1000000
     - Child 6: 1000001 to 1200000
   - The parent process must wait until all children have finished.

   Note: All processes must run concurrently.

   a) Provide a solution that meets the requirements of the problem.

   b) Is the output always sequential? Justify your answer.

   c) What changes to your program are needed to impose a sequential output?

   d) Run your program and, in another window, run several times the `ps` command. Comment on the results.

10. Implement a program to populate an array of 100,000 positions without repeating values. Then, spawn 5 new child processes. The objective is to determine, in a concurrent/parallel manner, whether a given number is present in the array.

    - Each child process handles 20000 positions.
    - The process that discovers the number should print the position where it was found and exit with its child number (1, 2, 3, 4, or 5).
    - Processes that do not find the number should exit with the value 0;
    - The parent process must wait for all children to terminate and print the PID and number of the child where the number was found, or an error message if the value was not found in the array.

    a) Provide a solution to the problem.

    b) What occurs when one process has found the number while the others are still running? Is it rational to continue using computer resources once the answer has been obtained?

11. Implement the function `char create_twins(pid_t list[2])`. This function creates two processes and returns the character `'a'` for the first child process and the character `'b'` for the second child. For the parent process, the function should return the character 'p'. In the parent process, the identifiers of the two created processes should be stored in the array passed as argument to the function.

    - Use the function `char create_twins(pid_t list[2])` to generate 6 new processes, each of which prints it's PID, it's parent's PID, and ends with the exit value `'a'` or `'b'`.
    - The parent process must wait for its children to terminate in the same order they were created and print their PIDs and exit values.

12. Consider the following program:

```
1  #include <stdio.h>
2  #include <unistd.h>
3  #include <errno.h>
4
5  int main(int argc,char *argv[]){
6     printf("Try to execute lls\n");
7     execl("/bin/lls","lls",NULL);
8     printf("execl returned! errno is [%d]\n",errno);
9     perror("The error message is :");
10    return 0;
11 }
```

    a) In this program, you can observe that the line "execl returned!..." is printed, indicating that `execl()` returns to the main function and the execution continues. Explain this behavior.

13. Consider the following program:

```
...
for(i=0;i<3;i++){
  pid=fork();

  if(pid==0)
    pid  =  fork();

  if(pid>0)
    execlp("scomp","scomp",NULL);
}
...
```

a) How many time is the `scomp` program executed? Justify your answer.

14. Consider the following program:

```
...
pid=fork();
pid=fork();

for(i=0;i<5;i++){
  pid=fork();

  execlp("exam","exam",NULL);

  if(pid==0)
    break;
}

execlp("students", "students","scomp",NULL);
...
```

a) How many times are the programs `exam` and `students` executed? Justify your answer.

15. Implement a program named `execute` that takes a set of Linux commands as argument to be executed. The program must run the commands concurrently and wait for their completion.

Example: `execute ps ls pwd`

a) Use the function `execlp` to solve the problem.

b) Use the function `execvp` to solve the problem.

c) In this scenario, which function is the most suitable? Justify your answer.

16. Implement a program that simulates a `shell`. The program only terminates when command "end" is entered. Optionally, add to your program the capability to accept command arguments, like in `ls -la`.

17. The `system` function enables the execution of a Linux command without creating another process using a `fork`. Investigate the usage of the `system` function and:

    a) Implement a simplified version of this function using `exec` functions and processes.

    b) Is the usage of the `system` function the most efficient way to execute external programs? Justify your answer.

18. Explain the result of the following `script` when we use an `exec` function to execute it.

```
#!/bin/cat -n
Computer Systems
```

19. Create a directory named "backup". Implement a program that asks the user for the name of a file and copies it to the "backup" directory.

20. An enthusiastic gambler wants to implement the "guess" program that simulates betting on a number. The rules are as follows:

    - The player selects a number between 1 and 5;
    - The program generates a random number between 1 and 5;
    - If the player correctly guesses the number at stake, he receives double the bet value.

    The random number is generated by a second program called "cheat". This program exits with a status between 1 and 5.

    The player starts with 25 euros and chooses the amount to bet on each round. The program ends when there is no more money to play.