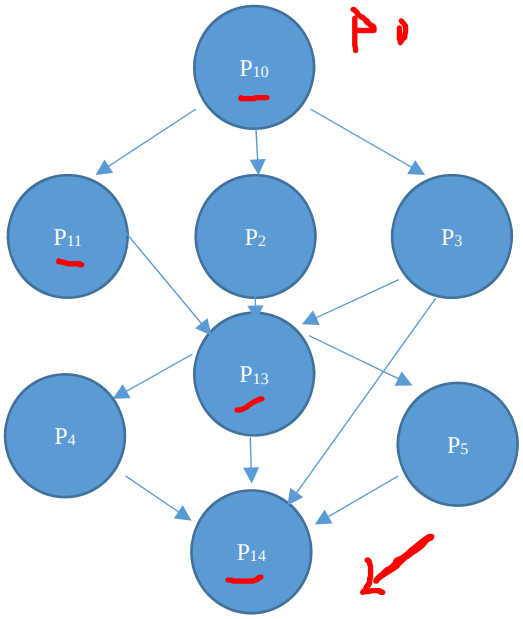
Operating Systems

Laboratory 4

Learning goals: In this laboratory activity, you will improve your understanding the use of `fork`, `wait`, and `waitpid` system calls for parent-child process synchronization.

Exercise 1

Write a C program that uses the system calls **`fork`** and **`waitpid`** to implement the following precedence graph, where labels P_{10}, P_{11}, P_{13}, and P_{14} represent sequential statements of the same process P_1.



The processes are, thus, five, and each statement or process in the precedence graph just prints its label and PID. Is this precedence graph feasible?

Process P_5 also asks the user to digit an 8-bit number that will be used as its return code.
Process P_{14} waits the termination of P_5 and prints the termination code of P_5.

Exercise 2

Using system calls **`fork`** and **`waitpid`** write a program that receive a number **`n`** from the command line and do the following:

1. creates **`n`** processes
2. the first created child sleeps **`n`** seconds, the second child sleeps **`n-1`** seconds, and so on.....(the last child sleeps **`1`** second)
3. the parent wait for "all" these children sequentially **from the first created to the last one.**

Exercise 3

Modify Exercise 2 using **`wait`** rather than **`waitpid`**. How many times the parent has to call **`wait`**? Does this program behaves as the previous one?