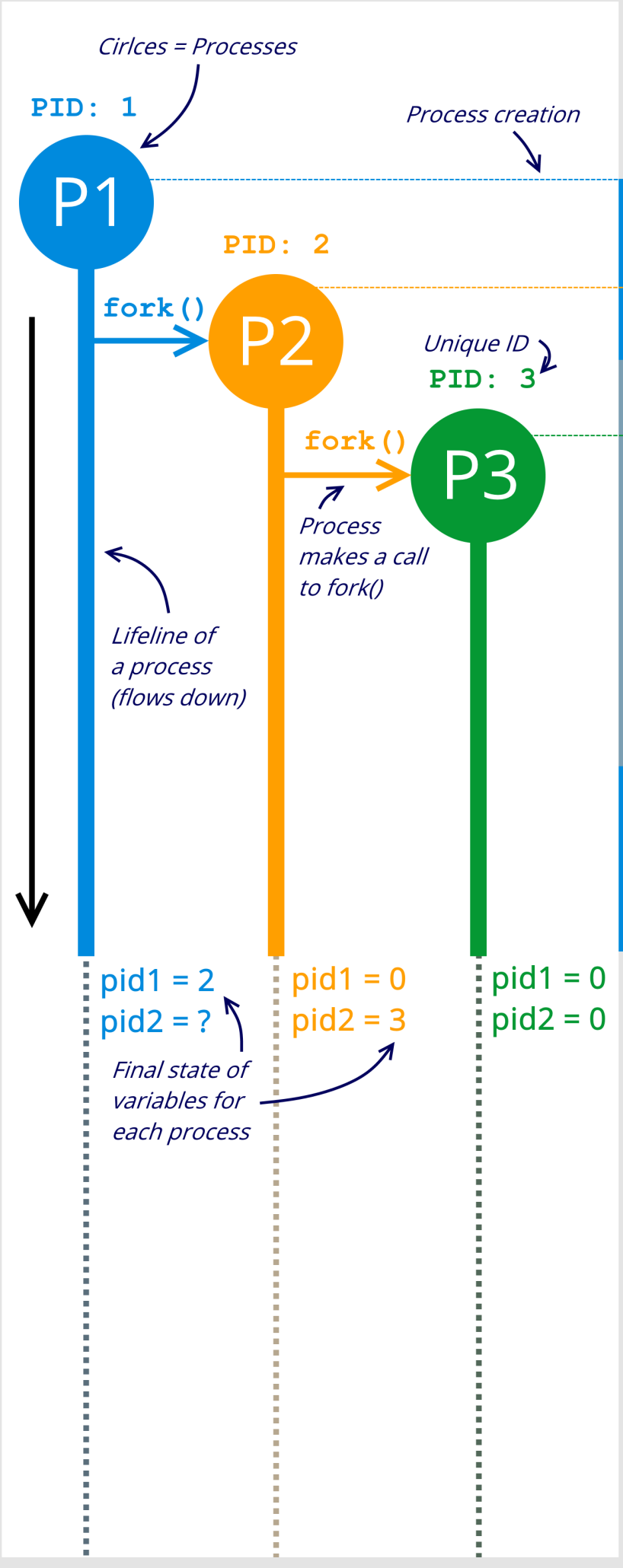


```
1  #include <sys/types.h>
2  #include <stdio.h>
3  #include <unistd.h>
4
5  int main() {
6      pid_t pid1, pid2;
7      pid1 = fork();
8
9      if (pid1 == 0) {
10         printf("A");
11         pid2 = fork();
12
13         if (pid2 == 0) {
14             printf("B");
15         }
16         else {
17             wait(NULL);
18             printf("C");
19         }
20     }
21     else {
22         printf("D");
23     }
24     return 0;
25 }
```

Lines of code touched per each process

Code Walkthrough

- Ln 5: P1 is created at the beginning of the program
- Ln 6: Create variables to hold a process ID (pid_t)
- Ln 7: Create a new process and store PID in 'pid1'
- Ln 9: Check which process is running
  - P1 and P2 are both running at this point, and the return value of fork is different for both processes. P1 will have the PID of the child process (P2) stored in 'pid1', whereas P2 will have 0 to show it's the child process.
- Ln 10: Print statement only executed by P2
- Ln 11: P2 spawns a child process and stores result in 'pid2'
- Ln 13: Check which process is running
  - P2 and P3 are both running, and the return values will be different for both. P2 stores the PID of the child process (P3) in 'pid2', whereas P3 has a 0 stored there to show it's the child process.
- Ln 14: Print statement only executed by P3
- Ln 16: If 'pid2' is not zero, then the parent process (P2) is running
- Ln 17: Wait for P3 to finish executing before moving on.
- Ln 18: Print statement only executed by P2
- Ln 21: If 'pid1' is not zero, then the parent process (P1) is running
- Ln 22: Print statement only executed by P1
- Ln 24: All processes return 0 and are subsequently terminated