fork() step by step...

Process 1000

```
int main(int argc, char * argv[])
{
    // Reached by parent process only (child doesn't exist yet)
    fprintf(stdout, "Hello form process %ld\n", (long) getpid());

    pid_t proc_id = fork();

    if(proc_id == -1) exit(1);      // Error...

    // Reached by both process (parent and child)
    fprintf(stdout, "Hello again form process %ld\n", (long) getpid());

    if(proc_id == 0) {
        // Reached by child process only
        fprintf(stdout, "CHILD  : PID=%ld\n", (long) getpid());
    } else {
        // Reached by parent process only
        fprintf(stdout, "PARENT : PID=%ld\n", (long) getpid());
    }

    // Reached by both process (parent and child)
    fprintf(stdout, "Goodbye form process %ld\n", (long) getpid());
}
```

fork() step by step...

Process 1000

```
int main(int argc, char * argv[])
{
    // Reached by parent process only (child doesn't exist yet)
    fprintf(stdout, "Hello form process %ld\n", (long) getpid());

    pid_t proc_id = fork();

    if(proc_id == -1) exit(1);      // Error...

    // Reached by both process (parent and child)
    fprintf(stdout, "Hello again form process %ld\n", (long) getpid());

    if(proc_id == 0) {
        // Reached by child process only
        fprintf(stdout, "CHILD  : PID=%ld\n", (long) getpid());
    } else {
        // Reached by parent process only
        fprintf(stdout, "PARENT : PID=%ld\n", (long) getpid());
    }

    // Reached by both process (parent and child)
    fprintf(stdout, "Goodbye form process %ld\n", (long) getpid());
}
```

Standard output (stdout):

```
Hello form process 1000
```

fork() step by step...

Process 1000 (PARENT)

```
int main(int argc, char * argv[])
{
    // Reached by parent process only (child doesn't exist yet)
    fprintf(stdout, "Hello form process %ld\n", (long) getpid());

    pid_t proc_id = fork();

    if(proc_id == -1) exit(1);      // Error...

    // Reached by both process (parent and child)
    fprintf(stdout, "Hello again form process %ld\n", (long) getpid());

    if(proc_id == 0) {
        // Reached by child process only
        fprintf(stdout, "CHILD  : PID=%ld\n", (long) getpid());
    } else {
        // Reached by parent process only
        fprintf(stdout, "PARENT : PID=%ld\n", (long) getpid());
    }

    // Reached by both process (parent and child)
    fprintf(stdout, "Goodbye form process %ld\n", (long) getpid());
}
```

Process 1001 (CHILD)

```
int main(int argc, char * argv[])
{
    // Reached by parent process only (child doesn't exist yet)
    fprintf(stdout, "Hello form process %ld\n", (long) getpid());

    pid_t proc_id = fork();

    if(proc_id == -1) exit(1);      // Error...

    // Reached by both process (parent and child)
    fprintf(stdout, "Hello again form process %ld\n", (long) getpid());

    if(proc_id == 0) {
        // Reached by child process only
        fprintf(stdout, "CHILD  : PID=%ld\n", (long) getpid());
    } else {
        // Reached by parent process only
        fprintf(stdout, "PARENT : PID=%ld\n", (long) getpid());
    }

    // Reached by both process (parent and child)
    fprintf(stdout, "Goodbye form process %ld\n", (long) getpid());
}
```

Standard output (stdout):

```
Hello form process 1000
```

fork() step by step...

Process 1000 (PARENT)

```
int main(int argc, char * argv[])
{
    // Reached by parent process only (child doesn't exist yet)
    fprintf(stdout, "Hello form process %ld\n", (long) getpid());

    pid_t proc_id = fork();

    if(proc_id == -1) exit(1);      // Error...

    // Reached by both process (parent and child)
    fprintf(stdout, "Hello again form process %ld\n", (long) getpid());

    if(proc_id == 0) {
        // Reached by child process only
        fprintf(stdout, "CHILD  : PID=%ld\n", (long) getpid());
    } else {
        // Reached by parent process only
        fprintf(stdout, "PARENT : PID=%ld\n", (long) getpid());
    }

    // Reached by both process (parent and child)
    fprintf(stdout, "Goodbye form process %ld\n", (long) getpid());
}
```

Process 1001 (CHILD)

```
int main(int argc, char * argv[])
{
    // Reached by parent process only (child doesn't exist yet)
    fprintf(stdout, "Hello form process %ld\n", (long) getpid());

    pid_t proc_id = fork();

    if(proc_id == -1) exit(1);      // Error...

    // Reached by both process (parent and child)
    fprintf(stdout, "Hello again form process %ld\n", (long) getpid());

    if(proc_id == 0) {
        // Reached by child process only
        fprintf(stdout, "CHILD  : PID=%ld\n", (long) getpid());
    } else {
        // Reached by parent process only
        fprintf(stdout, "PARENT : PID=%ld\n", (long) getpid());
    }

    // Reached by both process (parent and child)
    fprintf(stdout, "Goodbye form process %ld\n", (long) getpid());
}
```

Standard output (stdout):

```
Hello form process 1000
```

fork() step by step...

Process 1000 (PARENT)

```
int main(int argc, char * argv[])
{
    // Reached by parent process only (child doesn't exist yet)
    fprintf(stdout, "Hello form process %ld\n", (long) getpid());

    pid_t proc_id = fork();

    if(proc_id == -1) exit(1);      // Error...

    // Reached by both process (parent and child)
    fprintf(stdout, "Hello again form process %ld\n", (long) getpid());

    if(proc_id == 0) {
        // Reached by child process only
        fprintf(stdout, "CHILD  : PID=%ld\n", (long) getpid());
    } else {
        // Reached by parent process only
        fprintf(stdout, "PARENT : PID=%ld\n", (long) getpid());
    }

    // Reached by both process (parent and child)
    fprintf(stdout, "Goodbye form process %ld\n", (long) getpid());
}
```

Process 1001 (CHILD)

```
int main(int argc, char * argv[])
{
    // Reached by parent process only (child doesn't exist yet)
    fprintf(stdout, "Hello form process %ld\n", (long) getpid());

    pid_t proc_id = fork();

    if(proc_id == -1) exit(1);      // Error...

    // Reached by both process (parent and child)
    fprintf(stdout, "Hello again form process %ld\n", (long) getpid());

    if(proc_id == 0) {
        // Reached by child process only
        fprintf(stdout, "CHILD  : PID=%ld\n", (long) getpid());
    } else {
        // Reached by parent process only
        fprintf(stdout, "PARENT : PID=%ld\n", (long) getpid());
    }

    // Reached by both process (parent and child)
    fprintf(stdout, "Goodbye form process %ld\n", (long) getpid());
}
```

Standard output (stdout):

```
Hello form process 1000
Hello again form process 1000
```

fork() step by step...

Process 1000 (PARENT)

```
int main(int argc, char * argv[])
{
    // Reached by parent process only (child doesn't exist yet)
    fprintf(stdout, "Hello form process %ld\n", (long) getpid());

    pid_t proc_id = fork();

    if(proc_id == -1) exit(1);      // Error...

    // Reached by both process (parent and child)
    fprintf(stdout, "Hello again form process %ld\n", (long) getpid());

    if(proc_id == 0) {
        // Reached by child process only
        fprintf(stdout, "CHILD  : PID=%ld\n", (long) getpid());
    } else {
        // Reached by parent process only
        fprintf(stdout, "PARENT : PID=%ld\n", (long) getpid());
    }

    // Reached by both process (parent and child)
    fprintf(stdout, "Goodbye form process %ld\n", (long) getpid());
}
```

Process 1001 (CHILD)

```
int main(int argc, char * argv[])
{
    // Reached by parent process only (child doesn't exist yet)
    fprintf(stdout, "Hello form process %ld\n", (long) getpid());

    pid_t proc_id = fork();

    if(proc_id == -1) exit(1);      // Error...

    // Reached by both process (parent and child)
    fprintf(stdout, "Hello again form process %ld\n", (long) getpid());

    if(proc_id == 0) {
        // Reached by child process only
        fprintf(stdout, "CHILD  : PID=%ld\n", (long) getpid());
    } else {
        // Reached by parent process only
        fprintf(stdout, "PARENT : PID=%ld\n", (long) getpid());
    }

    // Reached by both process (parent and child)
    fprintf(stdout, "Goodbye form process %ld\n", (long) getpid());
}
```

Standard output (stdout):

```
Hello form process 1000
Hello again form process 1000
```

fork() step by step...

Process 1000 (PARENT)

```
int main(int argc, char * argv[])
{
    // Reached by parent process only (child doesn't exist yet)
    fprintf(stdout, "Hello form process %ld\n", (long) getpid());

    pid_t proc_id = fork();

    if(proc_id == -1) exit(1);      // Error...

    // Reached by both process (parent and child)
    fprintf(stdout, "Hello again form process %ld\n", (long) getpid());

    if(proc_id == 0) {
        // Reached by child process only
        fprintf(stdout, "CHILD  : PID=%ld\n", (long) getpid());
    } else {
        // Reached by parent process only
        fprintf(stdout, "PARENT : PID=%ld\n", (long) getpid());
    }

    // Reached by both process (parent and child)
    fprintf(stdout, "Goodbye form process %ld\n", (long) getpid());
}
```

Process 1001 (CHILD)

```
int main(int argc, char * argv[])
{
    // Reached by parent process only (child doesn't exist yet)
    fprintf(stdout, "Hello form process %ld\n", (long) getpid());

    pid_t proc_id = fork();

    if(proc_id == -1) exit(1);      // Error...

    // Reached by both process (parent and child)
    fprintf(stdout, "Hello again form process %ld\n", (long) getpid());

    if(proc_id == 0) {
        // Reached by child process only
        fprintf(stdout, "CHILD  : PID=%ld\n", (long) getpid());
    } else {
        // Reached by parent process only
        fprintf(stdout, "PARENT : PID=%ld\n", (long) getpid());
    }

    // Reached by both process (parent and child)
    fprintf(stdout, "Goodbye form process %ld\n", (long) getpid());
}
```

Standard output (stdout):

```
Hello form process 1000
Hello again form process 1000
```

fork() step by step...

Process 1000 (PARENT)

```
int main(int argc, char * argv[])
{
    // Reached by parent process only (child doesn't exist yet)
    fprintf(stdout, "Hello form process %ld\n", (long) getpid());

    pid_t proc_id = fork();

    if(proc_id == -1) exit(1);      // Error...

    // Reached by both process (parent and child)
    fprintf(stdout, "Hello again form process %ld\n", (long) getpid());

    if(proc_id == 0) {
        // Reached by child process only
        fprintf(stdout, "CHILD  : PID=%ld\n", (long) getpid());
    } else {
        // Reached by parent process only
        fprintf(stdout, "PARENT : PID=%ld\n", (long) getpid());
    }

    // Reached by both process (parent and child)
    fprintf(stdout, "Goodbye form process %ld\n", (long) getpid());
}
```

Process 1001 (CHILD)

```
int main(int argc, char * argv[])
{
    // Reached by parent process only (child doesn't exist yet)
    fprintf(stdout, "Hello form process %ld\n", (long) getpid());

    pid_t proc_id = fork();

    if(proc_id == -1) exit(1);      // Error...

    // Reached by both process (parent and child)
    fprintf(stdout, "Hello again form process %ld\n", (long) getpid());

    if(proc_id == 0) {
        // Reached by child process only
        fprintf(stdout, "CHILD  : PID=%ld\n", (long) getpid());
    } else {
        // Reached by parent process only
        fprintf(stdout, "PARENT : PID=%ld\n", (long) getpid());
    }

    // Reached by both process (parent and child)
    fprintf(stdout, "Goodbye form process %ld\n", (long) getpid());
}
```

Standard output (stdout):

```
Hello form process 1000
Hello again form process 1000
PARENT : PID=1000
```

fork() step by step...

Process 1000 (PARENT)

```
int main(int argc, char * argv[])
{
    // Reached by parent process only (child doesn't exist yet)
    fprintf(stdout, "Hello form process %ld\n", (long) getpid());

    pid_t proc_id = fork();

    if(proc_id == -1) exit(1);      // Error...

    // Reached by both process (parent and child)
    fprintf(stdout, "Hello again form process %ld\n", (long) getpid());

    if(proc_id == 0) {
        // Reached by child process only
        fprintf(stdout, "CHILD  : PID=%ld\n", (long) getpid());
    } else {
        // Reached by parent process only
        fprintf(stdout, "PARENT : PID=%ld\n", (long) getpid());
    }

    // Reached by both process (parent and child)
    fprintf(stdout, "Goodbye form process %ld\n", (long) getpid());
}
```

Process 1001 (CHILD)

```
int main(int argc, char * argv[])
{
    // Reached by parent process only (child doesn't exist yet)
    fprintf(stdout, "Hello form process %ld\n", (long) getpid());

    pid_t proc_id = fork();

    if(proc_id == -1) exit(1);      // Error...

    // Reached by both process (parent and child)
    fprintf(stdout, "Hello again form process %ld\n", (long) getpid());

    if(proc_id == 0) {
        // Reached by child process only
        fprintf(stdout, "CHILD  : PID=%ld\n", (long) getpid());
    } else {
        // Reached by parent process only
        fprintf(stdout, "PARENT : PID=%ld\n", (long) getpid());
    }

    // Reached by both process (parent and child)
    fprintf(stdout, "Goodbye form process %ld\n", (long) getpid());
}
```

Standard output (stdout):

```
Hello form process 1000
Hello again form process 1000
PARENT : PID=1000
Goodbye form process 1000
```

fork() step by step...

Process 1000 (PARENT)

```
int main(int argc, char * argv[])
{
    // Reached by parent process only (child doesn't exist yet)
    fprintf(stdout, "Hello form process %ld\n", (long) getpid());

    pid_t proc_id = fork();

    if(proc_id == -1) exit(1);      // Error...

    // Reached by both process (parent and child)
    fprintf(stdout, "Hello again form process %ld\n", (long) getpid());

    if(proc_id == 0) {
        // Reached by child process only
        fprintf(stdout, "CHILD  : PID=%ld\n", (long) getpid());
    } else {
        // Reached by parent process only
        fprintf(stdout, "PARENT : PID=%ld\n", (long) getpid());
    }

    // Reached by both process (parent and child)
    fprintf(stdout, "Goodbye form process %ld\n", (long) getpid());
}
```

Process 1001 (CHILD)

```
int main(int argc, char * argv[])
{
    // Reached by parent process only (child doesn't exist yet)
    fprintf(stdout, "Hello form process %ld\n", (long) getpid());

    pid_t proc_id = fork();

    if(proc_id == -1) exit(1);      // Error...

    // Reached by both process (parent and child)
    fprintf(stdout, "Hello again form process %ld\n", (long) getpid());

    if(proc_id == 0) {
        // Reached by child process only
        fprintf(stdout, "CHILD  : PID=%ld\n", (long) getpid());
    } else {
        // Reached by parent process only
        fprintf(stdout, "PARENT : PID=%ld\n", (long) getpid());
    }

    // Reached by both process (parent and child)
    fprintf(stdout, "Goodbye form process %ld\n", (long) getpid());
}
```

Standard output (stdout):

```
Hello form process 1000
Hello again form process 1000
PARENT : PID=1000
Goodbye form process 1000
```

fork() step by step...

Process 1000 (PARENT)

```
int main(int argc, char * argv[])
{
    // Reached by parent process only (child doesn't exist yet)
    fprintf(stdout, "Hello form process %ld\n", (long) getpid());

    pid_t proc_id = fork();

    if(proc_id == -1) exit(1);      // Error...

    // Reached by both process (parent and child)
    fprintf(stdout, "Hello again form process %ld\n", (long) getpid());

    if(proc_id == 0) {
        // Reached by child process only
        fprintf(stdout, "CHILD  : PID=%ld\n", (long) getpid());
    } else {
        // Reached by parent process only
        fprintf(stdout, "PARENT : PID=%ld\n", (long) getpid());
    }

    // Reached by both process (parent and child)
    fprintf(stdout, "Goodbye form process %ld\n", (long) getpid());
}
```

Process 1001 (CHILD)

```
int main(int argc, char * argv[])
{
    // Reached by parent process only (child doesn't exist yet)
    fprintf(stdout, "Hello form process %ld\n", (long) getpid());

    pid_t proc_id = fork();

    if(proc_id == -1) exit(1);      // Error...

    // Reached by both process (parent and child)
    fprintf(stdout, "Hello again form process %ld\n", (long) getpid());

    if(proc_id == 0) {
        // Reached by child process only
        fprintf(stdout, "CHILD  : PID=%ld\n", (long) getpid());
    } else {
        // Reached by parent process only
        fprintf(stdout, "PARENT : PID=%ld\n", (long) getpid());
    }

    // Reached by both process (parent and child)
    fprintf(stdout, "Goodbye form process %ld\n", (long) getpid());
}
```

Standard output (stdout):

```
Hello form process 1000
Hello again form process 1000
PARENT : PID=1000
Goodbye form process 1000
```

fork() step by step...

Process 1000 (PARENT)

```
int main(int argc, char * argv[])
{
    // Reached by parent process only (child doesn't exist yet)
    fprintf(stdout, "Hello form process %ld\n", (long) getpid());

    pid_t proc_id = fork();

    if(proc_id == -1) exit(1);      // Error...

    // Reached by both process (parent and child)
    fprintf(stdout, "Hello again form process %ld\n", (long) getpid());

    if(proc_id == 0) {
        // Reached by child process only
        fprintf(stdout, "CHILD  : PID=%ld\n", (long) getpid());
    } else {
        // Reached by parent process only
        fprintf(stdout, "PARENT : PID=%ld\n", (long) getpid());
    }

    // Reached by both process (parent and child)
    fprintf(stdout, "Goodbye form process %ld\n", (long) getpid());
}
```

Process 1001 (CHILD)

```
int main(int argc, char * argv[])
{
    // Reached by parent process only (child doesn't exist yet)
    fprintf(stdout, "Hello form process %ld\n", (long) getpid());

    pid_t proc_id = fork();

    if(proc_id == -1) exit(1);      // Error...

    // Reached by both process (parent and child)
    fprintf(stdout, "Hello again form process %ld\n", (long) getpid());

    if(proc_id == 0) {
        // Reached by child process only
        fprintf(stdout, "CHILD  : PID=%ld\n", (long) getpid());
    } else {
        // Reached by parent process only
        fprintf(stdout, "PARENT : PID=%ld\n", (long) getpid());
    }

    // Reached by both process (parent and child)
    fprintf(stdout, "Goodbye form process %ld\n", (long) getpid());
}
```

Standard output (stdout):

```
Hello form process 1000
Hello again form process 1000
PARENT : PID=1000
Goodbye form process 1000
Hello again form process 1001
```

fork() step by step...

Process 1000 (PARENT)

```
int main(int argc, char * argv[])
{
    // Reached by parent process only (child doesn't exist yet)
    fprintf(stdout, "Hello form process %ld\n", (long) getpid());

    pid_t proc_id = fork();

    if(proc_id == -1) exit(1);      // Error...

    // Reached by both process (parent and child)
    fprintf(stdout, "Hello again form process %ld\n", (long) getpid());

    if(proc_id == 0) {
        // Reached by child process only
        fprintf(stdout, "CHILD  : PID=%ld\n", (long) getpid());
    } else {
        // Reached by parent process only
        fprintf(stdout, "PARENT : PID=%ld\n", (long) getpid());
    }

    // Reached by both process (parent and child)
    fprintf(stdout, "Goodbye form process %ld\n", (long) getpid());
}
```

Process 1001 (CHILD)

```
int main(int argc, char * argv[])
{
    // Reached by parent process only (child doesn't exist yet)
    fprintf(stdout, "Hello form process %ld\n", (long) getpid());

    pid_t proc_id = fork();

    if(proc_id == -1) exit(1);      // Error...

    // Reached by both process (parent and child)
    fprintf(stdout, "Hello again form process %ld\n", (long) getpid());

    if(proc_id == 0) {
        // Reached by child process only
        fprintf(stdout, "CHILD  : PID=%ld\n", (long) getpid());
    } else {
        // Reached by parent process only
        fprintf(stdout, "PARENT : PID=%ld\n", (long) getpid());
    }

    // Reached by both process (parent and child)
    fprintf(stdout, "Goodbye form process %ld\n", (long) getpid());
}
```

Standard output (stdout):

```
Hello form process 1000
Hello again form process 1000
PARENT : PID=1000
Goodbye form process 1000
Hello again form process 1001
```

fork() step by step...

Process 1000 (PARENT)

```
int main(int argc, char * argv[])
{
    // Reached by parent process only (child doesn't exist yet)
    fprintf(stdout, "Hello form process %ld\n", (long) getpid());

    pid_t proc_id = fork();

    if(proc_id == -1) exit(1);      // Error...

    // Reached by both process (parent and child)
    fprintf(stdout, "Hello again form process %ld\n", (long) getpid());

    if(proc_id == 0) {
        // Reached by child process only
        fprintf(stdout, "CHILD  : PID=%ld\n", (long) getpid());
    } else {
        // Reached by parent process only
        fprintf(stdout, "PARENT : PID=%ld\n", (long) getpid());
    }

    // Reached by both process (parent and child)
    fprintf(stdout, "Goodbye form process %ld\n", (long) getpid());
}
```

Process 1001 (CHILD)

```
int main(int argc, char * argv[])
{
    // Reached by parent process only (child doesn't exist yet)
    fprintf(stdout, "Hello form process %ld\n", (long) getpid());

    pid_t proc_id = fork();

    if(proc_id == -1) exit(1);      // Error...

    // Reached by both process (parent and child)
    fprintf(stdout, "Hello again form process %ld\n", (long) getpid());

    if(proc_id == 0) {
        // Reached by child process only
        fprintf(stdout, "CHILD  : PID=%ld\n", (long) getpid());
    } else {
        // Reached by parent process only
        fprintf(stdout, "PARENT : PID=%ld\n", (long) getpid());
    }

    // Reached by both process (parent and child)
    fprintf(stdout, "Goodbye form process %ld\n", (long) getpid());
}
```

Standard output (stdout):

```
Hello form process 1000
Hello again form process 1000
PARENT : PID=1000
Goodbye form process 1000
Hello again form process 1001
CHILD : PID=1001
```

fork() step by step...

Process 1000 (PARENT)

```
int main(int argc, char * argv[])
{
    // Reached by parent process only (child doesn't exist yet)
    fprintf(stdout, "Hello form process %ld\n", (long) getpid());

    pid_t proc_id = fork();

    if(proc_id == -1) exit(1);      // Error...

    // Reached by both process (parent and child)
    fprintf(stdout, "Hello again form process %ld\n", (long) getpid());

    if(proc_id == 0) {
        // Reached by child process only
        fprintf(stdout, "CHILD  : PID=%ld\n", (long) getpid());
    } else {
        // Reached by parent process only
        fprintf(stdout, "PARENT : PID=%ld\n", (long) getpid());
    }

    // Reached by both process (parent and child)
    fprintf(stdout, "Goodbye form process %ld\n", (long) getpid());
}
```

Process 1001 (CHILD)

```
int main(int argc, char * argv[])
{
    // Reached by parent process only (child doesn't exist yet)
    fprintf(stdout, "Hello form process %ld\n", (long) getpid());

    pid_t proc_id = fork();

    if(proc_id == -1) exit(1);      // Error...

    // Reached by both process (parent and child)
    fprintf(stdout, "Hello again form process %ld\n", (long) getpid());

    if(proc_id == 0) {
        // Reached by child process only
        fprintf(stdout, "CHILD  : PID=%ld\n", (long) getpid());
    } else {
        // Reached by parent process only
        fprintf(stdout, "PARENT : PID=%ld\n", (long) getpid());
    }

    // Reached by both process (parent and child)
    fprintf(stdout, "Goodbye form process %ld\n", (long) getpid());
}
```

Standard output (stdout):

```
Hello form process 1000
Hello again form process 1000
PARENT : PID=1000
Goodbye form process 1000
Hello again form process 1001
CHILD : PID=1001
```

fork() step by step...

Process 1000 (PARENT)

```
int main(int argc, char * argv[])
{
    // Reached by parent process only (child doesn't exist yet)
    fprintf(stdout, "Hello form process %ld\n", (long) getpid());

    pid_t proc_id = fork();

    if(proc_id == -1) exit(1);      // Error...

    // Reached by both process (parent and child)
    fprintf(stdout, "Hello again form process %ld\n", (long) getpid());

    if(proc_id == 0) {
        // Reached by child process only
        fprintf(stdout, "CHILD  : PID=%ld\n", (long) getpid());
    } else {
        // Reached by parent process only
        fprintf(stdout, "PARENT : PID=%ld\n", (long) getpid());
    }

    // Reached by both process (parent and child)
    fprintf(stdout, "Goodbye form process %ld\n", (long) getpid());
}
```

Process 1001 (CHILD)

```
int main(int argc, char * argv[])
{
    // Reached by parent process only (child doesn't exist yet)
    fprintf(stdout, "Hello form process %ld\n", (long) getpid());

    pid_t proc_id = fork();

    if(proc_id == -1) exit(1);      // Error...

    // Reached by both process (parent and child)
    fprintf(stdout, "Hello again form process %ld\n", (long) getpid());

    if(proc_id == 0) {
        // Reached by child process only
        fprintf(stdout, "CHILD  : PID=%ld\n", (long) getpid());
    } else {
        // Reached by parent process only
        fprintf(stdout, "PARENT : PID=%ld\n", (long) getpid());
    }

    // Reached by both process (parent and child)
    fprintf(stdout, "Goodbye form process %ld\n", (long) getpid());
}
```

Standard output (stdout):

```
Hello form process 1000
Hello again form process 1000
PARENT : PID=1000
Goodbye form process 1000
Hello again form process 1001
CHILD : PID=1001
Goodbye form process 1001
```

fork() step by step...

Child **Parent**

```
int main(int argc, char * argv[])
{
    // Reached by parent process only (child doesn't exist yet)
    fprintf(stdout, "Hello form process %ld\n", (long) getpid());

    pid_t proc_id = fork();

    if(proc_id == -1) exit(1);      // Error...

    // Reached by both process (parent and child)
    fprintf(stdout, "Hello again form process %ld\n", (long) getpid());

    if(proc_id == 0) {
        // Reached by child process only
        fprintf(stdout, "CHILD  : PID=%ld\n", (long) getpid());
    } else {
        // Reached by parent process only
        fprintf(stdout, "PARENT : PID=%ld\n", (long) getpid());
    }

    // Reached by both process (parent and child)
    fprintf(stdout, "Goodbye form process %ld\n", (long) getpid());
}
```