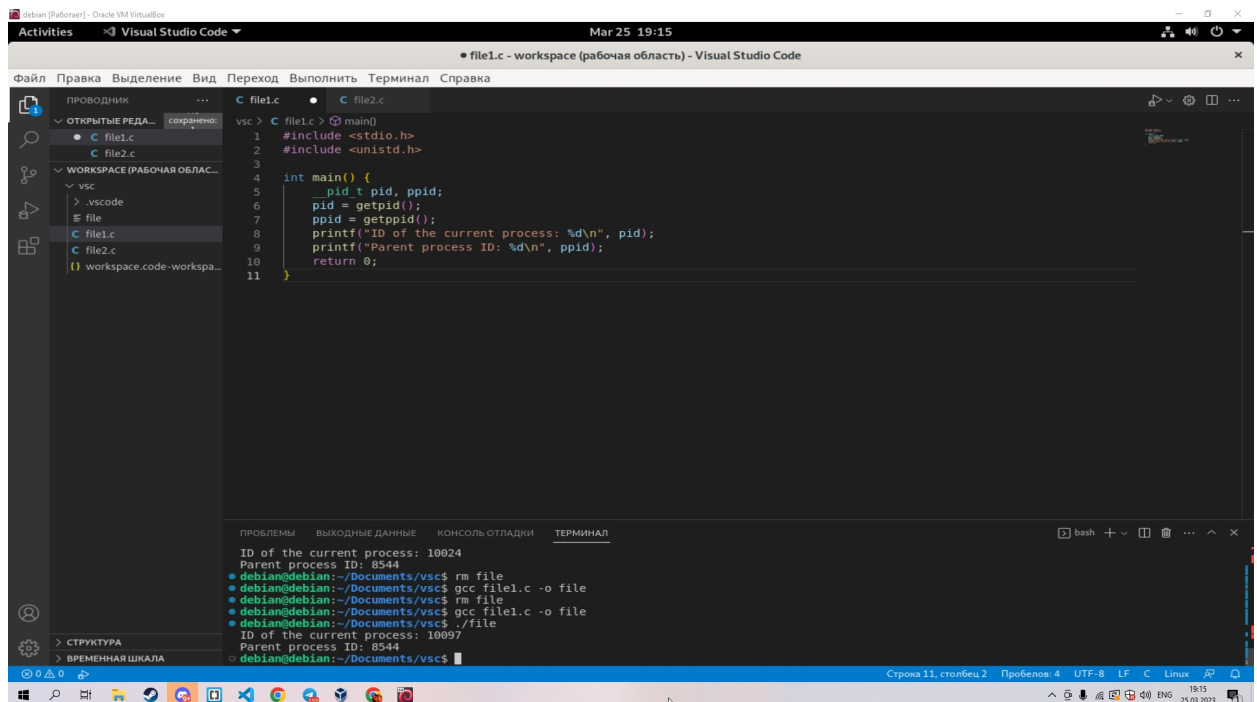


1) Выводит идентификатор текущего и родительского процесса

Функция `getpid()` возвращает идентификатор текущего процесса, а функция `getppid()` возвращает идентификатор родительского процесса. Обе функции определены в заголовочном файле `<unistd.h>`. Затем мы выводим полученные идентификаторы с помощью функции `printf()`.



The screenshot shows the Visual Studio Code interface with a C file named `file1.c` open. The code is as follows:

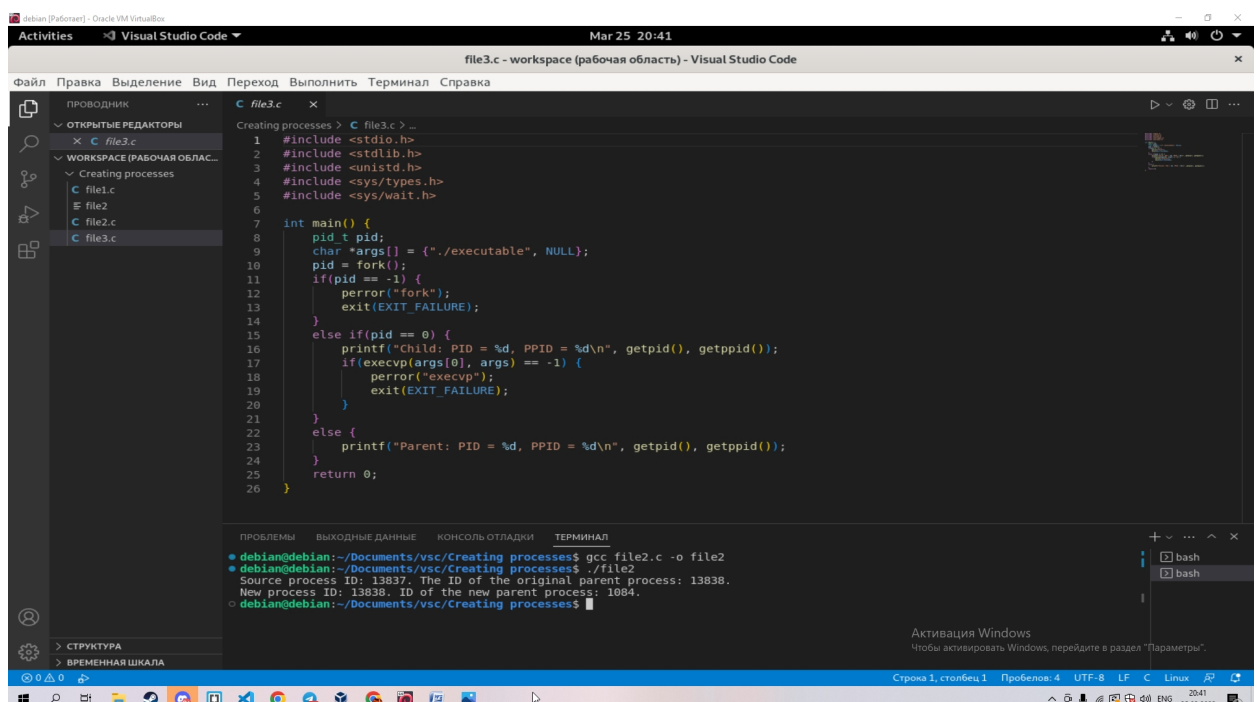
```
1 #include <stdio.h>
2 #include <unistd.h>
3
4 int main() {
5     _pid_t pid, ppid;
6     pid = getpid();
7     ppid = getppid();
8     printf("ID of the current process: %d\n", pid);
9     printf("Parent process ID: %d\n", ppid);
10    return 0;
11 }
```

The terminal output shows the execution of the program:

```
debian@debian:~/Documents/vsc$ gcc file1.c -o file
debian@debian:~/Documents/vsc$ ./file
ID of the current process: 10024
Parent process ID: 8544
debian@debian:~/Documents/vsc$ rm file
debian@debian:~/Documents/vsc$ gcc file1.c -o file
debian@debian:~/Documents/vsc$ ./file
ID of the current process: 10097
Parent process ID: 8544
debian@debian:~/Documents/vsc$
```

2) Создаёт копии самой себя и выводит идентификатор текущего и родительского процесса (видно, что идентификаторы меняются)

Для создания копии процесса используется системный вызов `fork()`. Чтобы получить идентификатор текущего процесса, используется системный вызов `getpid()`. Для получения идентификатора родительского процесса, используется системный вызов `getppid()`.



The screenshot shows the Visual Studio Code interface with a C file named `file3.c` open. The code is as follows:

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <unistd.h>
4 #include <sys/types.h>
5 #include <sys/wait.h>
6
7 int main() {
8     pid_t pid;
9     char *args[] = {"/.executable", NULL};
10    pid = fork();
11    if (pid == -1) {
12        perror("fork");
13        exit(EXIT_FAILURE);
14    }
15    else if (pid == 0) {
16        printf("Child: PID = %d, PPID = %d\n", getpid(), getppid());
17        if (execvp(args[0], args) == -1) {
18            perror("execvp");
19            exit(EXIT_FAILURE);
20        }
21    }
22    else {
23        printf("Parent: PID = %d, PPID = %d\n", getpid(), getppid());
24    }
25    return 0;
26 }
```

The terminal output shows the execution of the program:

```
debian@debian:~/Documents/vsc/Creating processes$ gcc file2.c -o file2
debian@debian:~/Documents/vsc/Creating processes$ ./file2
Source process ID: 13837. The ID of the original parent process: 13838.
New process ID: 13838. ID of the new parent process: 1084.
debian@debian:~/Documents/vsc/Creating processes$
```

3) реализация программы из пункта 2 с помощью создания процесса с новым кодом (вызов exes)

Данная программа создает новый процесс с помощью функции `fork()`, а затем вызывает новый процесс с помощью функции `execvp()`

```
file3.c - workspace (рабочая область) - Visual Studio Code

Файл  Правка  Выделение  Вид  Переход  Выполнить  Терминал  Справка

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

C file3.c
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <unistd.h>
4 #include <sys/types.h>
5 #include <sys/wait.h>
6
7 int main() {
8     pid_t pid;
9     char *args[] = {"/.executable", NULL};
10    pid = fork();
11    if(pid == -1) {
12        perror("fork");
13        exit(EXIT_FAILURE);
14    }
15    else if(pid == 0) {
16        printf("Child: PID = %d, PPID = %d\n", getpid(), getppid());
17        if(execvp(args[0], args) == -1) {
18            perror("execvp");
19            exit(EXIT_FAILURE);
20        }
21    }
22    else {
23        printf("Parent: PID = %d, PPID = %d\n", getpid(), getppid());
24    }
25    return 0;
26 }
```

```
debian@debian:~/Documents/vsc/Creating processes$ ./file2
Source process ID: 13837. The ID of the original parent process: 13838.
New process ID: 13838. ID of the new parent process: 1084.
debian@debian:~/Documents/vsc/Creating processes$ rm file2
debian@debian:~/Documents/vsc/Creating processes$ gcc file3.c -o file3
debian@debian:~/Documents/vsc/Creating processes$ ./file3
Parent: PID = 13924, PPID = 13745
Child: PID = 13925, PPID = 1084
execvp: No such file or directory
debian@debian:~/Documents/vsc/Creating processes$
```

Ссылка на репозиторий: <https://github.com/Volodin-Ilya/operating-systems.git>