**Семинар 8.**

Системный вызов fork, создающий точную копию процесса. Затем вместо копии мы можем заменить образ приложения с помощью вызова execve.

Если же нам нужно просто сделать копию родительского процесса, которая ведёт себя немного по-другому, execve даже не нужен.

-------------------------------------------------------------------------------------------------------------------

**1.Вопрос** разберите этот исходный файл.

**1.1.Вопрос** что именно возвращает fork() в родительском процессе? А что в дочернем? См. man fork.

#include <unistd.h>

#include <stdio.h>

int main() {

// pid\_t это, как и int, численный тип для идентификаторов процессов

pid\_t pid = fork();

// в зависимости от возвращаемого значения, которое попадёт в pid, мы поймём, находимся ли мы в родительском процессе или в дочернем.

if (pid == 0)

{

printf("I am a child, pid was %d\n", pid);

}

else

{

printf("I am a parent, pid was %d\n", pid);

}

}

Fork() создаёт точную копию процесса.

Возвращаемое значение в родительском процессе (вызывающий форк)- PID дочернего процесса.

Возвращаемое значение в процессе-клоне(дочернем)-0. Поэтому это и проверяется в коде выше.

Если по какой-то причине клонировать процесс не удалось, то родителю возвращается -1, а ребёнку ничего не вернется, так как его нет, ведь клонировать не удалось.