2 more ways to know if a process exists:

*One way is the **getpgid(pid_t pid)** function:*

getpgid() returns the PGID(process group id) of the process specified by pid. If pid is zero, the process ID of the calling process is used.
On error, -1 is returned, and errno is set appropriately.(similar to the kill function).
ESRCH: pid does not match any process.
That way we can know if the process exists or not.

The advantage of getpgid() function:

- We get the PGID. This is important information for times we will want to send signals to the process's group.

The disadvantage of getpgid() function :

- We can check just if any process exists without the ability to cause a process to execute certain commands (send signals).
  This is compared to the **kill** function through which we can send signals to process in order to execute certain commands.

*Second way is the **stat** function:*

Use this function in the following way:
int stat(const char *pathname, struct stat *statbuf);
These functions return information about a file, in the buffer pointed to by statbuf.
The proc filesystem is a pseudo-filesystem which provides an interface to kernel data structures. It is commonly mounted at /proc.
In our program, we need to pass the /proc/<pid> path to the function as pathname argument.
The fields of the statbuf argument are filled in by the stat function.
This structure holds meta data on the file sent to the function.
If the return value of the function is zero, then the pathname is exists. This make us to be sure that the process with the given pid is exists.
On error, -1 is returned, and errno is set appropriately.
If errno is ENOENT → A component of pathname does not exist. → the process with the given pid is not exists.

The advantage of stat function:

- We get more information about the process besides his status.

The disadvantage of stat function:

- We can not use this method in Windows because the proc folder is not exists there.
- We can check just if any process exists without the ability to cause a process to execute certain commands (send signals).
  This is compared to the **kill** function through which we can send signals to process in order to execute certain commands.

Third way is the kill function: ***kill(pid,sig)***

In our program we used the “kill” function with “zero signal”
If sig is zero, then no signal is sent, but existence and permission checks are still performed.
This can be used to check for the existence of a process ID or process group ID that the caller is permitted to signal.
We can know that with the value result of function. (if the value result is zero it’s means that there is process with pid value that exists).

The advantage of kill function:

- The kill function returns a detailed answer in case there is an error by errno.
- We can use this function to send the process another signal after we sure that the process exists and have permission. We can actually send signals!

The disadvantage of kill function:

- Some signals may be ignored (like our program).
- With “zero signal” we can check just if any process exists without to see more information about the process or more information about the process’s group as we can see in the **stat** function and **getpgid** function.

The common disadvantage of all this functions is that we must know the pid of process.