1. Fork

Library : unistd

return value : 0 - returned in the child

: -1 - fail, returned in the parent ⟩ succesful

: PID > 0 - returned in the parent

2. Wait

Libraries: # include <sys/types.h>

include <sys/wait.h>

Return value:

- on succes : - returns the process ID of the terminated child
  - on error -1 is returned.

exec() family of functions:
replaces the current process image with a new process image
Library: #include <unistd.h>
it only returns if there was an error
$\hookrightarrow -1$

does it for you unlike exec ↗
System(C) - creates a child process and let's you run shell code in c

$$\begin{tikzcd}
C = NULL \arrow[r, leftarrow] \arrow[dr] & \neq 0 \text{ shell available} \\
 & = 0 \text{ shell not available}
\end{tikzcd}$$

$$\begin{tikzcd}
C \neq NULL \arrow[r, leftarrow] \arrow[dr] & -1 \text{ er. case} \\
 & \text{termination status of command}
\end{tikzcd}$$

ex: System("ls")

signal (s, funct)

s - int

funct - void (*f)(int)

ex:
```
   #include <signal.h>
  void f (int s){
      printf ("print something");
      return
  }

  int main(){
      signal (SIGINT, f);
      while (1);
      return 0;
  }
```

man 7 signal

Kill (int PID, int S)

Library: # include <signal.h>

Return: $\begin{cases} 0 - \text{if successfull} \\ -1 - \text{there is an error} \end{cases}$