wait()

The `wait()` function suspends execution of the calling process until one of its childs terminates. If a child has already exited, returns immediately with the child's termination status.

It's defined in the "sys/wait.h" system header.

```
#include <sys/types.h>  // defines pid_t
#include <sys/wait.h>

// "wstatus" is a pointer to the exit status
//          of the terminated child
pid_t wait(int *wstatus);
```

If "wstatus" is not NULL, can be analyzed using the following macros (defined in "sys/wait.h"):

- WIFEXITED(wstatus) → Evaluates to true if child exited normally
- WEXITSTATUS(wstatus) → Returns child's exit code
- WIFSIGNALED(wstatus) → Evaluates to true if child was terminated by a signal
- WTERMSIG(wstatus) → Returns the signal number that caused the child to terminate

If "wstatus" is NULL, the exit status is not stored.

The function returns the PID of the terminated child, if success.

In case of failure, returns -1 and sets "errno".

Example Usage

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/wait.h>
#include <unistd.h>

int main(void)
{
    pid_t pid;
    int status;

        pid = fork();
    if (pid == 0)  // child process
    {
        printf("Child process\n");
        exit(42);
    }
    else  // parent process
    {
        wait(&status);
        if (WIFEXITED(status))
            printf("Child exited with status %d\n", WEXITSTATUS(status));
    }
    return (0);
}
```