wait4()

The `wait4()` function is a more flexible version of “waitpid()”, defined in the “sys/wait.h” system header.

- Can wait for a specific child, or any child depending on “pid”
- Supports non-blocking waits using “options” flags like “WNOHANG”
- Provides resource usage stats with “struct rusage”
- Available in Linux via glibc, unlike “wait3()”, it’s commonly used in Linux

```
#include <sys/types.h> // includes "pid_t"
#include <sys/resource.h>  // includes "struct rusage"
#include <sys/wait.h>

// "pid" determines which children to wait for
//     - "< -1" wait for any child whose PGID equals abs(pid)
//     - "-1" wait for any child process
//     - "0" wait for any child in the same process group (PGID)
//     - "> 0" wait for a specific child with that PID
// "wstatus" gets child exit status
// "options" configuration flags to modify behavior
// "rusage" collects cpu and memory usage
pid_t wait4(pid_t pid, int *wstatus, int options, struct rusage *rusage);
```

Example Usage

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

int main(void)
{
    pid_t pid;
    int status;
    struct rusage usage;

        pid = fork();
    if (pid == 0)  // child
    {
        for (volatile int i = 0; i < 100000000; ++i);
        exit(5);
    }
    else
    {
        wait4(pid, &status, 0, &usage);
        if (WIFEXITED(status))
        {
            printf("Child exited with %d\n", WEXITSTATUS(status));
            printf("User time: %ld.%06ld\n", usage.ru_utime.tv_sec, usage.ru_utime.tv
        }
    }
    return (0);
}
```