fork()

The fork() system call causes the creation of a new process. The new process (the child process) is an exact copy of the calling process (the parent process).

```
SYNOPSIS
     #include <unistd.h>
      pid_t fork(void);

DESCRIPTION
   The child process inherits the following attributes from the
parent process ... (man)
   The child process differs from the parent process in the
following ways:
```

- The child process has a unique process ID.
- The child process ID does not match any active process group ID.
- The child process has a different parent process ID (which is the process ID of the parent process).
- The set of signals pending for the child process is initialized
- to the empty set.

fork()

`pid_t` is the generic process type. **Under Unix, this is a `short`.**
`fork()` can only return three things:

- **`0`** for a child process
- **`-1`** no child was created
- **`else`** PID of your child is returned to the parent

Child

`exit()`
When the child calls `exit()`, the return value passed will arrive at the parent when it `wait()`s

Parent

`wait()` it waits for whichever one happens to exit first
`waitpid()` specify exactly which child to wait

fork()

```
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>
int main(void)
{
pid_t pid;
int rv;
switch(pid = fork()) {
case -1:
   perror("fork"); /* something went wrong */
   exit(1); /* parent exits */
case 0:
   printf(" CHILD: This is the child process!\n");
   printf(" CHILD: My PID is %d\n", getpid());
   printf(" CHILD: My parent's PID is %d\n", getppid());
   printf(" CHILD: Enter my exit status (make it small): ");
   scanf(" %d", &rv);
   printf(" CHILD: I'm outta here!\n");
   exit(rv);
```

fork()

```
default:
   printf("PARENT: This is the parent process!\n");
   printf("PARENT: My PID is %d\n", getpid());
   printf("PARENT: My child's PID is %d\n", pid);
   printf("PARENT: I'm now waiting for my child to exit()...\n");
   wait(&rv);
   printf("PARENT: My child's exit status is: %d\n", WEXITSTATUS(rv));
   printf("PARENT: I'm outta here!\n");
}
return 0;
}
```

fork()

```
if (!fork()) {
    printf("I'm the child!\n");
    exit(0);
} else {
    printf("I'm the parent!\n");
    wait(NULL);
}
```

```
main()
{
signal(SIGCHLD, SIG_IGN); /* now I don't have to wait()! */
.
.
fork();fork();fork(); /* Rabbits, rabbits, rabbits! */
}
```