pipe()

The `pipe()` function creates a unidirectional Inter-Process Communication (IPC) channel, allowing data to flow between processes. It's defined in the "unistd.h" system header.

It populates two file descriptors:

- "pipefd[0]" → Read end (data reception)
- "pipefd[1]" → Write end (data transmission)

Data written to "pipefd[1]" can be read from "pipefd[0]" in FIFO order.

```
#include <unistd.h>

// "pipefd" is an array to store file descriptors
int pipe(int pipefd[2]);
```

Returns 0 on success. In case of failure, returns -1 and sets "errno".

- Data flows only from "pipefd[1]" to "pipefd[0]"
- It's buffered by the kernel. Default buffer is typically 64KB
- Reads block if the pipe is empty, writes block if the pipe is full

Example Usage

```
int main(void) {
    int pipefd[2];
    char buf[20];
    pid_t pid;
    if (pipe(pipefd) == -1) {
        perror("pipe");
        return (1);
    }
    pid = fork();
    if (pid == -1) {
        perror("fork");
        return (1);
    }
    if (pid == 0)  // child process (writer)
    {  // close unused read end
        close(pipefd[0]);
        write(pipefd[1], "Hello, parent!", 14);
        close(pipefd[1]);
    } else  // parent process (reader)
    {  // close unused write end
        close(pipefd[1]);
        read(pipefd[0], buf, sizeof(buf));
        printf("Parent received: %s\n", buf);
        close(pipefd[0]);
        // wait for child
        wait(NULL);
    }
    return (0);
}
```