# Tee program
# with many files output

Mikolaj Baczynski

# The Project description

Program reads stdin and writes to all output files given as arguments.

Program can be run in three possible ways:

- without any options – program creates new files if any file exists program ends with error
- with option -a – program appends content to existing files, or creates if any doesn’t exist
- with option -t – program creates new files if any file exists program deletes content and write to the file.

# The challenge

- Save current stdin(content) without ending program
- ‘\n’ doesn’t stop reading stdin
- Handle case when the buffer is full
- Handle case when error occurred while program reading from “big file”.

# The Idea

I use buffer to holding the content and I fill it with read() function. When the buffer is full program automatically saves content into file and starts reading input again.

Read() exits when ‘meets’ ‘\n’, but also the number of bytes read is returned. So I keep filling the buffer but from buffer + bytes_read.

CTRL+D sends EOF. I use it to save the content.

CTRL+C (SIGINT) ends program but if read() doesn’t finish reading the function will be restarted.

# The solution

1

```
void tee(int nfile, char *files[], int flag){
    size_t BUFSIZ = 8192;
    if(write(STDOUT_FILENO, "To save your work to file(s)
        press CTRL + D,\nTo exit press CTRL C\n", 67) < 0)
        perror("Error - writing on STDOUT ");
    char buffer[BUFSIZ];
    int bytes_read = 0;
    int temp = 1;
```

Sigaction Config

Restart interupted function

```
struct sigaction sa;
sa.sa_flags = 0;
sa.sa_flags = SA_RESTART;
sigemptyset(&sa.sa_mask);
sa.sa_handler = handler;
if(sigaction(SIGINT, &sa, NULL)<0)
    perror("Error - sigction");
```

2

Global var – is updated when SIGINT occured

```
while (1){
        if (stop == 1)
            _exit(0);

        temp = read( STDIN_FILENO, buffer + bytes_read, sizeof buffer -
bytes_read );
        if(temp < 0 || errno == EINTR)
            perror("Error - reading from STDIO ");
        else if (temp != 0){
            bytes_read += temp;
        }
        else{
            for (short i = 0; i < nfile; ++i){
                    int file = open(files[i], flag, 0777);
                    if (file == -1){
                        perror("Error - opening file");
                    }
                    else{
                        if(write(file, buffer, bytes_read)<0)
                            perror("Error - writing to file ");
                        if(close(file) < 0)
                            perror("Error - closing the file ");
                    }
                }
            temp = 1;
            bytes_read = 0;
            flag = (O_WRONLY | O_APPEND | O_CREAT);
        }
    }
}
```

If \n keep reading

Update Flags

# The summary

The project wasn't difficult. I’ve learned the basics of I/O, signals.

Unsolved, I have not figured out when a file is blocking by another program.

# Thank You