fpurge() hack — Koszek trick #1

W. Adam Koszek

Koszek ORG

wojciech@koszek.com

2022-10-05T17:47:25Z

A friend of mine asked me once:

How to recover text formatted by printf(), but do not spit it output to the console?

Right now I don't exactly remember why `fprintf()` couldn't be used, but I remember this was in the environment where `fprintf()` (or all?) changes couldn't be applied.

This is a technique I was came up with. Please note: this is *HUGE* hack and it is not proven to work on your system. It is based on the fact data from `printf` function family is first written to the buffer `buf`, and until it's internally flushed, it won't be printed. Lack of flush is implicitly expected due to the large buffer size, and buffering set by `setbuf`. Once we generate enough data in the `buf` buffer through `fprintf` calls, we copy the data to `text` buffer, and discard `buf` buffer. Discarding happens via `fprune` and its associated file descriptor, for which buffer has been used for.

It used to work on FreeBSD and GNU/Linux, but PLEASE don't make your code depend on that. It's a hack. Here it is:

```
#include <assert.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define SZ 1024*1024

/*
 * This is Koszek's trick to write formated data to the FILE structure
 * with an underlying buffer, but without flushing it to the backing
```

```
 * descriptor. Later on, once a buffer is filled, I print it to get a
 * proof and clear it.
 */
int
main(int argc, char **argv)
{
 char buf[SZ];
 char text[SZ];
 FILE *fp;

 (void)argc;
 (void)argv;

 memset(buf, 0, sizeof(buf));

 fp = fopen("FILE.txt", "a+");
 assert(fp != NULL);
 fflush(fp);
 setbuf(fp, buf);

 fprintf(fp, "Nie wiem co napisaæ\n");
 fprintf(fp, "Nie wiem co tutaj wklepaæ\n");
 fprintf(fp, "To ma tylko wyczy¶ciæ bufor\n");

 memcpy(text, buf, sizeof(text));

 fpurge(fp);
 setbuf(fp, NULL);

 printf("Text: '%s'\n", text);

 fflush(fp);
 fclose(fp);
 return (EXIT_SUCCESS);
}
```