Lab1上机回忆

先说下结果，exam很简单顺利通过，extra鏖战100min得了25分。（最后割须弃袍，落荒而逃，还把水杯落在了机房

这是一场 披着OS外衣的 程设 C8 上机，考察的全是指针。题型和 2023 年的完全一致，可参考以下两位学长的博客：

buaa nr

cookedbear

exam

题面

实现一个自定义格式化字符串 `%[flags][width][length]k`，功能是输出一个键值对 `<key> => <value>`

- `<key>`、`=>`、`<value>`之间固定要有一个空格
- `<key>` 是字符串类型，格式遵从 `[flags]` 和 `[width]]`
- `<value>` 是 `int` 或 `long int` 类型，格式遵从 `[flags]` 和 `[width]]`

样例输入

```
printk("%k","num",123);
```

样例输出

```
num => 123
```

挺简单的，复制 `%...d`、`%...s`、`%...c` 的实现即可

我的代码

```
case 'k':
            s = (char *)va_arg(ap, char *);
            print_str(out, data, s, width, ladjust);

            c = ' ';
            print_char(out, data, c, 1, ladjust);
            c = '=';
            print_char(out, data, c, 1, ladjust);
            c = '>';
            print_char(out, data, c, 1, ladjust);
            c = ' ';
            print_char(out, data, c, 1, ladjust);

            if (long_flag)
            {
```

```
            num = va_arg(ap, long int);
        }
        else
        {
            num = va_arg(ap, int);
        }

        if (num < 0)
        {
            num = -num;
            neg_flag = 1;
        }
        print_num(out, data, num, 10, neg_flag, width, ladjust, padc, 0);

        break;
```

extra

其实这个也不难，就是实现一个 类似 `sprintf` 的函数，两位学长的博客都已提到过。但是指针雪藏多年实在生疏，导致出现了漫山遍野的 `Bug`。

题面

实现一个简单的内存文件流（`fmemopen`、`fmemprintf`、`fseek`、`fclose`），类似于标准库的 `FILE*` 操作，但针对内存缓冲区

- 首先是我们的 FILE 结构体类型

```
typedef struct{
    char *ptr;          // 写指针
    char *base;         // 基指针
    char *end;          // 尾指针
}FILE;
```

- 实现函数 `FILE *fmemopen(FILE *stream, void *buf, const char *mode);`
  - 功能：根据打开方式 `mode`，初始化内存流 `FILE *stream`
  - 若 `mode` 为 `"w"`，则为覆盖式写入。令 `stream` 的写指针、基指针、尾指针都指向 `buf`。返回参数 `stream`
  - 若 `mode` 为 `"a"`，则为追加式写入。令 `stream` 的写指针指向 `buf` 中第一个 `'\0'`，基指针、尾指针都指向 `buf`。返回参数 `stream`
  - 若 `mode` 不为以上二者，则返回 `NULL`
- 实现函数 `int fmemprintf(FILE *stream, const char *fmt, ...);`
  - 功能：将格式化字符串写入内存流 `FILE *stream`，并 **返回写入的字符数**
  - 提示：仿照 `printk` 函数实现，可以调用你课下完成的 `vprintfmt` 函数。

`void vprintfmt(fmt_callback_t out, void *data, const char *fmt, va_list ap)` 的前两个参数被称为 回调函数、回调上下文。简单来说，就是通过调用 `out` 函数，将格式化字符串输出到 `data` 中。而课下作业是输出到控制台，回调函数 `outputk` 并没有用到 `data`，故 `data` 没有起到作用。

对于要求实现的 `fmemprintf` 函数，你可以传入内存流 `stream` 作为 `data`，并且仿照 `outputk` 写出合适的回调函数

- 实现函数 `int fseek(FILE *stream, long offset, int fromwhere);`
  - 功能：将写指针挪到指定位置
  - 若 `fromwhere` 为 `SEEK_SET`（宏），则写指针指向 `stream->base + offset`
  - 若 `fromwhere` 为 `SEEK_CUR`（宏），则写指针指向 `stream->ptr + offset`
  - 若 `fromwhere` 为 `SEEK_END`（宏），则写指针指向 `stream->end + offset`
  - 若 `fromwhere` 为以上三种取值，且挪动后的写指针位于闭区间 `[ base , end ]` 中，则返回 0；否则返回 -1
- 实现函数 `int fclose(FILE *stream)`
  - 关闭内存流
  - 将尾指针指向的内容置为 `'\0'`

提供待补全的 `string.c`，其中已实现 `memcpy`、`memset`、`strcpy`、`strlen`、`strchr`、`strcmp` 函数。

首先你需要引入所需的头文件：

```
#include <print.h>
#include <stream.h>
```

然后在 `string.c` 末尾实现上述4个函数

样例输入

```
暂无
```

样例输出

```
暂无
```

我的代码

```
FILE *fmemopen(FILE *stream, void *buf, const char *mode){
    if(strcmp(mode,"w")==0){
        stream->ptr = (char*)buf;
        stream->base = (char*)buf;
        stream->base = (char*)buf;
    }
    else if (strcmp(mode,"a")==0){
        stream->base = (char*)buf;
        char  *s = (char*)strchr(buf,'\0');
```

```
        stream->ptr = s;
        stream->end = s;
    }
    else{
        return NULL;
    }
    return stream;
}

void outputfile(FILE *data,const char *buf,size_t len){

         char *s = data->ptr;
         for(int i = 0; i<len ; i++){
                 *s = buf[i];
                 s++;
         }
         data->ptr = s;
         data->end = s;
 }

int fmemprintf(FILE *stream, const char *fmt, ...){
    char *ptr1 = stream->ptr;
    va_list ap;
    va_start(ap,fmt);
    vprintfmt(outputfile,stream,fmt,ap);
    va_end(ap);
    char *ptr2 = stream->ptr;
    int len = ptr2-ptr1;
    return len;
}

int fseek(FILE *stream, long offset, int fromwhere){
    char *s = stream->ptr;
    char *base = stream->base;
    char *end = stream->end;
    s = s + (fromwhere + offset);

    if ((s-base > 0)&&(end-s > 0)){
        stream->ptr = s;
        return 0;

    }
    else{
        return -1;
    }

}

int fclose(FILE *stream){
    char *end = stream->end;
    *end = '\0';
    return 0;
}
```

Bug分析

- `void*`、`char*`、`const char*`这几种指针要 **强制转换**，否则会编译错误
- 注意 `fmemopen` 中，判断 `mode` 时**不能 只解引用来判断第一个字符**

心得体会

菜就多练😭