Assignment3

- Code

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

typedef struct memstruct {
	char *buf;
	size_t bufsize;
	size_t curpos;
} memstruct;

FILE* myfmemopen(void*, size_t);
int myread(void*, char*, int);
int mywrite(void*, const char*, int);
fpos_t myseek(void*, fpos_t, int);
int myclose(void*);

FILE* myfmemopen(void* buf, size_t size) {

	memstruct *mem;
	mem = malloc(sizeof(memstruct));

	mem -> buf = buf;
	mem -> bufsize = size;
	mem -> curpos = 0;

	FILE* fp = funopen(mem, myread, mywrite, myseek, myclose);
	return fp;
}

int myread(void* cookie, char* buf, int nbytes) {
	memstruct* mem = cookie;
	int maxlen;
	if (nbytes > mem -> bufsize - mem -> curpos)
		maxlen = mem -> bufsize - mem -> curpos;
	else
		maxlen = nbytes;
	memcpy(buf, mem -> buf + mem -> curpos, maxlen);
	mem -> curpos += maxlen;
	return maxlen;
}

int mywrite(void* cookie, const char* buf, int nbytes) {
	memstruct* mem = cookie;
	memcpy(mem -> buf + mem -> curpos, buf, nbytes);
	return nbytes;
}

fpos_t myseek(void* cookie, fpos_t pos, int whence) {
	memstruct* mem = cookie;
	fpos_t res;

	switch(whence) {
		case SEEK_SET:
			res = pos;
			break;
		case SEEK_END:
			res = mem -> bufsize + pos;
			break;
		case SEEK_CUR:
			res = mem -> curpos;
			break;
		default:
			break;
	}
	mem -> curpos = res;
	return res;
}

int myclose(void* cookie) {
	memstruct* mem = cookie;
```

```
71          free(mem -> buf);
72          free(mem);
73          return 0;
74  }
75
76  int main() {
77          const char s[] = "hello, world";
78          char buf[1024] = {0};
79          FILE* fp = myfmemopen(buf, 1024);
80          char buffer[1024];
81          if (fwrite(s, sizeof(char), 12, fp) +1 != sizeof(s))
82                  fprintf(stderr, "fwrite error\n");
83          if (fseek(fp, 7L, SEEK_SET))
84                  fprintf(stderr, "fseek error\n");
85          fread(buffer, sizeof(char), sizeof(buffer), fp);
86          puts(buffer);
87          if(fseek(fp, 0L, SEEK_SET))
88                  fprintf(stderr, "fseek error\n");
89          fread(buffer, sizeof(char), sizeof(buffer), fp);
90          puts(buffer);
91          fclose(fp);
92          return 0;
93  }
94
```

- Line 5 to Line 9: (memstruct definition)
  建立儲存 file stream 的資料型態 memstruct, 其中包含字串、字串長度及當前指向的位置。
- Line 11 to Line 15:
  宣告 fmemopen 及 funopen 會用到的四個函數 read, write, seek, close. 這裡在每個宣告的函數加上 'my' 字首以避免編譯時報錯，例如若將 myfmemopen 宣告為 fmemopen, 在 FreeBSD 的 gcc 下會報錯。
- Line 17 to Line 28: (myfmemopen implementation)
  開啟一個 file stream, 預設大小為 1024 個 characters. 每個值皆為 0, 且指向位置為 0. 最後呼叫 funopen, 將自定義的 read, write, lseek, close 傳入 fp 並回傳 fp.
- Line 30 to Line 40: (myread implementation)
  將 mem 當前被指向的位置(curpos)往後讀 min(nbytes, 最多能讀的數量) 個字元，並將讀取到的字串寫進 buf 中。最後再將 curpos 移動到剛剛讀取時的最後一個位置。
- Line 42 to Line 46: (mywrite implementation)
  將 buf 內的字串 append 到 curpos 後面。
- Line 48 to Line 67: (myseek implementation)
  先判斷要從 mem 的開頭、當前指向位置(curpos)或是結尾，再將其值加上 offset.(這裡取名為pos) 最後再將 curpos 指向最後移動到的位置並回傳其值。
- Line 69 to Line 74: (myclose implementation)
  釋放記憶體，除了 mem 本身要釋放以外，mem 中的 buf 也要一起釋放。

- Line 79:
  指定 fp 為透過我們自己實作的 fmemopen 回傳的 FILE pointer.
- Line 81 to Line 82:
  將 hello, world 寫進 fp, 若成功寫入的字元數和預期要寫入的不同則報錯。
- Line 83 to Line 84:
  移動 curpos 到 w 的位置。若回傳值不為 0 則報錯。
- Line 85:
  將 FILE stream 裡讀取 curpos 之後的所有字元，並盡可能將 buffer 塞滿。在這裡，我們就是把 world 讀進 buffer 後，file stream 就沒有其他東西可以讀了。
- Line 86:
  將 buffer 的內容輸出。
- Line 87 to Line 90:
  和上面類似。由於 file stream 內仍有 hello, world 整組字串，因此我們同樣透過調整 curpos, 讀進 buffer 後再將 buffer 內的東西輸出。
- Line 91:
  將 fp 釋放。