Universität Hamburg, Fachbereich Informatik
Arbeitsbereich Wissenschaftliches Rechnen
E. Betke, N. Hübbe, M. Kuhn, J. Lüttgau, J. Squar

Übungsblatt 11 zum Praktikum
C-Programmierung
im WiSe 2019/2020
Abgabe: 19.01.2020, 23:59

1 Stückweises Lesen einer Datei

Implementieren Sie `read_file` im folgenden Programm so, dass der übergebene Dateideskriptor `fd` vollständig gelesen wird, wobei Ihnen nur ein einzelner Integer (`buf`) als Puffer zur Verfügung steht. Erweitern Sie die Funktion um Code, der immer einen einzelnen Integer-Wert liest, bis er am Ende der Datei angekommen ist. Werte ungleich `0` sollen dabei zusammen mit ihrem Offset in der Datei ausgegeben werden.

```c
#include <assert.h>
#include <fcntl.h>
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>

void read_file(int fd)
{
    int buf;

    // TODO
    printf("int=%d at offset=%lu\n", buf, 0);
}

int main(void)
{
    int fd;

    fd = open("myfile", O_RDWR | O_CREAT, 0600);
    assert(fd != -1);
    assert(pwrite(fd, &fd, sizeof(fd), 1048576) == sizeof(fd));
    read_file(fd);
    assert(pwrite(fd, &fd, sizeof(fd), 1024) == sizeof(fd));
    read_file(fd);
    assert(pwrite(fd, &fd, sizeof(fd), 524288) == sizeof(fd));
    read_file(fd);
    assert(close(fd) == 0);
    assert(unlink("myfile") == 0);

    return 0;
}
```

2 Persistente Datenstrukturen (30 Punkte)

Das folgende Programm implementiert eine Array-Datenstruktur, die in einem per `malloc` allokierten Speicherbereich verwaltet wird. Passen Sie das Programm so an, dass die Array-Datenstruktur mithilfe von `mmap` in einer Datei `myarray` verwaltet wird. `array_init` soll dabei den vorherigen Zustand aus der Datei wiederherstellen, sofern sie bereits vorhanden ist. Die restlichen Funktionen sollen sich wie bisher verhalten.

```
1  #include <fcntl.h>
2  #include <stdio.h>
3  #include <stdlib.h>
4  #include <string.h>
5  #include <sys/mman.h>
6  #include <sys/types.h>
7  #include <sys/stat.h>
8  #include <unistd.h>
9
10 struct array_element
11 {
12     int age;
13     char name[256];
14 };
15
16 struct array
17 {
18     off_t size;
19     off_t count;
20     struct array_element element[];
21 };
22
23 void array_init(struct array* arr, off_t size)
24 {
25     arr->size = size;
26     arr->count = 0;
27 }
28
29 void array_reset(struct array* arr)
30 {
31     arr->count = 0;
32 }
33
34 void array_append(struct array* arr, int age, char* name)
35 {
36     if ((sizeof(*arr) + sizeof(struct array_element) *
       ↪ (arr->count + 1)) > arr->size)
37     {
38         return;
39     }
40
41     arr->element[arr->count].age = age;
42     strncpy(arr->element[arr->count].name, name, 256);
```

```
43      arr->count++;
44  }
45
46  struct array_element* array_get(struct array* arr, off_t index)
47  {
48      if (index >= arr->count)
49      {
50          return NULL;
51      }
52
53      return &arr->element[index];
54  }
55
56  void array_print(struct array* arr)
57  {
58      for (off_t i = 0; i < arr->count; i++)
59      {
60          struct array_element* e = array_get(arr, i);
61          printf("arr[%lu] = { %d, %s }\n", i, e->age, e->name);
62      }
63  }
64
65  int main(void)
66  {
67      struct array* arr;
68
69      arr = malloc(1024);
70
71      array_init(arr, 1024);
72      array_print(arr);
73
74      array_reset(arr);
75      array_append(arr, 18, "Max Mustermann");
76      array_append(arr, 21, "Moritz Mustermann");
77      array_append(arr, 18, "Petra Mustermann");
78      array_append(arr, 21, "Paula Mustermann");
79      array_print(arr);
80
81      array_reset(arr);
82      array_append(arr, 21, "Paula Mustermann");
83      array_print(arr);
84
85      free(arr);
86
87      return 0;
88  }
```

Abgabe

Abzugeben ist ein gemäß den bekannten Richtlinien erstelltes und benanntes Archiv. Das enthaltene und gewohnt benannte Verzeichnis soll folgenden Inhalt haben:

- Alle Quellen, aus denen Ihr Programm besteht (`exercise1.c`); gut dokumentiert (Kommentare im Code!)

Senden Sie Ihre Abgabe an `cp-abgabe@wr.informatik.uni-hamburg.de`.