Informatik und Angewandte Kognitionswissenschaft
Lehrstuhl für Hochleistungsrechnen

Thomas Fogal
Prof. Dr. Jens Krüger

High-Performance Computing

http://hpc.uni-due.de/teaching/wt2014/nbody.html

Exercise 1 (15 Points)

All assignments should be pushed to your personal Git repository. Assignments are due at midnight on the due date. No late assignments will be accepted.

All assignments must include a `Makefile` for compiling your assignments. Assignments which do not compile will receive 0 points. Assignments that do not satisfy the test inputs will receive 0 points.

Please **do not include output** other than what was requested by the assignment details.

Your assignment will be graded on the `duecray.uni-due.de` supercomputer. It does not matter if your program runs correctly on another machine; it must run correctly on `duecray` to receive credit.

In this assignment you will practice with C pointers, learn how to work with files, and manage dynamic memory.

1 Reading and sorting an array (10 points)

Your program will read an array of integers from a file and sort those integers. The sorted list must then be output. You will need to use pointers extensively, as the sort happens in-place.

Your program must accept two arguments: the first will be the name of a file containing a set of (ASCII) integers. The second is the number of integers that exist in the file. The program must allocate an array large enough to

hold the data, read the data from the file into the array, sort it, and output the sorted array. Listing 1 shows a few sample runs of a correct program.

```
$ cat small
123
932
20
$ ./sort small 3
{ 20 123 932 }
$ ./sort small 2
{ 123 932 }
$ ./sort small 1
{ 123 }
```

Listing 1: Sample execution of the program.

Your makefile should, by default, produce an executable named exactly `sort`, as in Listing 1.

1.1 File format

The file simply contains a set of integers, separated by whitespace. Note that both:

```
1 2      3
4
```

as well as:

```
1
2
3


4
```

are both valid inputs and should result in the same output. That is, whether the whitespace is a newline or other spacing is unimportant. You should use the `" %d"` format specifier.

In contrast to the previous assignment, you will need to open and close a file. Additionally, instead of reading from standard input, you must read from the file. The additional functions you will need are `fopen`, `fscanf`, and `fclose`.

1.2 Starting code

To help get you started, Listing 2 gives a few functions that you should use in your assignment. Note that `swap` is unimplemented and `sort` is missing a line.

```
/* Swap two values using their addresses. */
static void
swap(int* e1, int* e2)
{
  /* todo: implement  me! */
}

/* give the index of the minimum value in the array. */
static size_t
idx_minimum(int* arr, size_t n)
{
  if(n == 0) { abort(); }
  size_t idx = 0;
  for(size_t i=1; i < n; ++i) {
    if(arr[i] < arr[idx]) { idx = i; }
  }
  return idx;
}

/* an in-place sort of the given array. */
static void
sort(int* arr, size_t n)
{
  for(size_t i=0; i < n; ++i) {
    /* 'i' is the current index.  find the minimum and swap with that. */
    const size_t minidx = idx_minimum(arr+i, n-i) + i;
    /* todo: this is missing a statement: add that in! */
  }
}
```

Listing 2: Sample functions to use in your assignment.

Hint: implement *and test* `swap` first, and be sure you have it right before moving on to the `sort`.

1.3 Other new functions

You will also need to make use of `malloc` and `free` to allocate and deallocate the array. The argument for the number of values to read is, like all arguments, given as a string. You can use the `atoi` function to convert it into an integer.

2 EMail me an explanation of how sort and swap work (5 points)

Send me an email that explains how the `sort` and `swap` functions work, especially the interaction between the two.

Additionally, include a question or comment concerning something about the course or HPC in general.

This does not have to be long: a paragraph and a sentence is fine.

3 *sine qua non*

As always, submit your assignment by committing your code to your personal repository. Every assignment utilizes a new repository. The name for this assignment is **as1-*username***, where *username* is your name on our git server.

All assignments must include a makefile to compile your program. No makefile, no points!