The Design of *OrionOS* Operating System

Students' Name

November 2, 2013

Abstract

A short description of this report content.

Take into account that a design document is a complete high-level solution to the problem presented. It should be detailed enough that somebody who already understands the problem could go out and code the project without having to make any significant decisions. Further, if this somebody happens to be an experienced coder, they should be able to use the design document to code the solution in a few hours (not necessarily including debugging).

Chapter 1

General Presentation

1.1 Working Team

Specify the working team members' names and their responsibility.

1. Pană Alexandru
   (a) Threads: dealt with AAA
   (b) Threads: dealt with BBB
   (c) Threads: dealt with CCC
2. Soucup Adrian
   (a) Threads: dealt with AAA
   (b) Threads: dealt with BBB
   (c) Threads: dealt with CCC
3. Vultur Horațiu
   (a) Threads: dealt with AAA
   (b) Threads: dealt with BBB
   (c) Threads: dealt with CCC
4. Zene Andrei

(a) Threads: dealt with AAA

(b) Threads: dealt with BBB

(c) Threads: dealt with CCC

Chapter 2

Design of Module *threads*

2.1 Initial Functionality

Describe briefly what you have been given (have started from) at the beginning of the assignment and the way the existing functionality must be extended.

2.2 Data Structures and Functions

Specify the *data structures* and *functions* involved in your solution and what are they used for. Describe in few words the purpose of new added data structures, fields and functions.

```
struct thread
{

      ...
  /* the time (in number of ticks) at which a
      sleeping thread should wake up */
      int64_t wakeup_time;
      ...

};
```

```
/* a sorted list containing all sleeping
(blocked) threads. Sorting is done using
thread_wakeup_time_comparison function*/
static struct list sleep_list;

/* comparison function to order the sleeping
threads ascending by their wakeup time */
list_less_func thread_wakeup_time_comparison;

/* sets the time at which the thread should
wake up and puts the thread in the sleeping
threads list */
void thread_sleep (int64_t);

/*this function checks if there are sleeping
threads that should wake up at this moment
and calls the function thread_unblock for
each of those threads.*/
void handle_sleeping_threads();
```

2.3 Functionality

Describe **briefly**, still **explicitly**, in words and pseudocode the way your solution works. DO NOT INCLUDE detailed code.

Give examples, if you think they can make your explanation clearer. You are free to use any other techniques (e.g. use-case diagrams, sequence diagrams etc.) that you think can make your explanation clearer. See Figure 2.1 below to see the way images are inserted in a Latex file.

Here you have a pseudo-code description of an algorithm taken fom http://en.wikibooks.org/wiki/LaTeX. It uses the *program* package. Alternatively, you can use *algorithmic* or *algorithm2e* packages.

Example of a pseudo-code algorithm description:

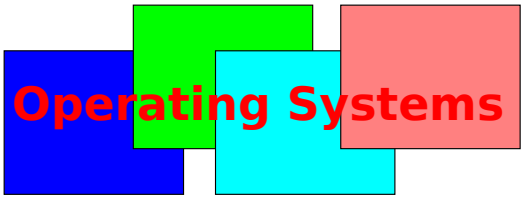

Figure 2.1: Sample image

```
begin for i := 1 to 10 step 1 do
            expt(2, i);
            newline() od          This text will be set flush to the right margin
where
proc expt(x, n) ≡
  z := 1;
  do if n = 0 then exit fi;
     do if odd(n) then exit fi;
        comment: This is a comment statement;
        n := n/2; x := x * x od;
     {n > 0};
     n := n − 1; z := z * x od;
  print(z).
end
```

2.4 Design Decisions

Justify your design decisions, specify other design alternatives, their advantages and disadvantages and mention the reasons of your choice.

2.5 Tests

Describe briefly the tests you are intended to run in order to test the functionality of your implementation.

2.6 Observations

You can use this section to mention other things not mentioned in the other sections.

You can indicate and evaluate, for instance:

- the most difficult parts of your assignment and the reasons you think they were so;
- the difficulty level of the assignment and if the allocated time was enough or not;
- particular facts or hints you think we should students to help them solve better the assignment.

You can also make suggestions for teacher, relative to the way he can assist more effectively the students.

Chapter 3

Design of Module *userprog*

3.1 Initial Functionality

Describe briefly what you have been given (have started from) at the beginning of the assignment and the way the existing functionality must be extended.

3.2 Data Structures and Functions

Specify the *data structures* and *functions* involved in your solution and what are they used for. Describe in few words the purpose of new added data structures, fields and functions.

```
struct existent_data_structure {
        int newField;
};

struct newDataStructure {
};

int newFunction();
```

3.3 Functionality

Describe **briefly**, still **explicitly**, in words and pseudocode the way your solution works. DO NOT INCLUDE detailed code.

Give examples, if you think they can make your explanation clearer. You are free to use any other techniques (e.g. use-case diagrams, sequence diagrams etc.) that you think can make your explanation clearer.

3.4 Design Decisions

Justify your design decisions, specify other design alternatives, their advantages and disadvantages and mention the reasons of your choice.

3.5 Tests

Describe briefly the tests you are intended to run in order to test the functionality of your implementation.

3.6 Observations

You can use this section to mention other things not mentioned in the other sections.

You can indicate and evaluate, for instance:

- the most difficult parts of your assignment and the reasons you think they were so;
- the difficulty level of the assignment and if the allocated time was enough or not;
- particular facts or hints you think we should give students to help them solve better the assignment.

You can also make suggestions for teacher, relative to the way he can assist more effectively the students.

Chapter 4

Design of Module *virtualmemory*

4.1 Initial Functionality

Describe briefly what you have been given (have started from) at the beginning of the assignment and the way the existing functionality must be extended.

4.2 Data Structures and Functions

Specify the *data structures* and *functions* involved in your solution and what are they used for. Describe in few words the purpose of new added data structures, fields and functions.

```
struct existent_data_structure {
        int newField;
};

struct newDataStructure {
};

int newFunction();
```

4.3 Functionality

Describe **briefly**, still **explicitly**, in words and pseudocode the way your solution works. DO NOT INCLUDE detailed code.

Give examples, if you think they can make your explanation clearer. You are free to use any other techniques (e.g. use-case diagrams, sequence diagrams etc.) that you think can make your explanation clearer.

4.4 Design Decisions

Justify your design decisions, specify other design alternatives, their advantages and disadvantages and mention the reasons of your choice.

4.5 Tests

Describe briefly the tests you are intended to run in order to test the functionality of your implementation.

4.6 Observations

You can use this section to mention other things not mentioned in the other sections.

You can indicate and evaluate, for instance:

- the most difficult parts of your assignment and the reasons you think they were so;
- the difficulty level of the assignment and if the allocated time was enough or not;
- particular facts or hints you think we should give students to help them solve better the assignment.

You can also make suggestions for teacher, relative to the way he can assist more effectively the students.