Lectures

- In the lecture on February, 9th you will get a C programming introduction.
- In the lecture on February, 12th we will finish the discussion of System Structures (Chapter 2).
- The 1st can be found on the course homepage after the Monday lecture, the 2nd mandatory assignment will be put on the course homepage also during week 07. Note, that there are neither lectures nor exercises in week 08 and 09, therefore you should focus on these assignment in these 2 weeks.
- Voluntary reading for DTrace:
  - Dynamic Instrumentation of Production Systems, Bryan M. Cantrill, Michael W. Shapiro and Adam H. Leventhal, Solaris Kernel Development Sun Microsystems `http://learningsolaris.com/docs/dtrace_usenix.pdf`
  - DTrace Intro :`https://wikis.oracle.com/display/DTrace/Introduction` and `http://www.brendangregg.com/dtrace.html`
  - The examples that were shown in the lecture can be downloaded here: `http://www.imada.sdu.dk/~daniel/DM510-2015/add/dm510-dtrace-examples.tar.gz`
- Voluntary reading for SystemTap: `https://sourceware.org/systemtap/`
- Voluntary reading on Linux kernel profiling with perf: `https://perf.wiki.kernel.org/index.php/Tutorial`
- Note: The slides used in the lecture for the second chapter differ significantly from the original slides provided by Wiley.

Tutorial Session

- Prepare for the Tutorial Session on Tuesday, February 11, 2015:

  Chapter 1:
  1.1, 1.3, 1.4, 1.5, 1.7, 1.8, 1.9, 1.10, 1.11, 1.14, 1.19

  Chapter 2:
  2.2, 2.4, 2.5, 2.6, 2.7, 2.9, 2.10, 2.11

  C programming exercises
  You are expected to already have C programming skills to a certain degree. In case you need to improve your C programming skills, you can for example follow the online C tutorial `http://www.cprogramming.com/tutorial.html#ctutorial`. More links can be found on the homepage of the course. In the tutorial session focus on pointers and the subsequent exercise which will prepare you for the 2nd mandatory assignment.

1. Which of the following is the proper declaration of a pointer?
   - `int x;`
   - `int &x;`
   - `ptr x;`
   - `int *x;`
2. Which of the following gives the memory address of integer variable a?
   - `*a;`
   - `a;`
   - `&a;`
   - `address(a);`
3. Which of the following gives the memory address of a variable pointed to by pointer a?
   - `a;`
   - `*a;`
   - `&a;`
   - `address(a);`
4. Which of the following gives the value stored at the address pointed to by pointer a?
   - `a;`
   - `val(a);`
   - `*a;`
   - `&a;`
5. Which of the following is the proper keyword or function to allocate memory in C?
   - `new`
   - `malloc`
   - `create`
   - `value`
6. Which of the following is the proper keyword or function to deallocate memory?
   - `free`
   - `delete`
   - `clear`
   - `remove`

Analyse the following C source code. Discuss what it does.

```
#include "dm510_msgbox.h"
#include <stdlib.h>
#include <string.h>

typedef struct _msg_t msg_t;

struct _msg_t{
  msg_t* previous;
  int length;
  char* message;
};

static msg_t *bottom = NULL;
static msg_t *top = NULL;

int dm510_msgbox_put( char *buffer, int length ) {
  msg_t* msg = malloc(sizeof(msg_t));
  msg->previous = NULL;
  msg->length = length;
  msg->message = malloc(length);
  memcpy(msg->message, buffer, length);

  if (bottom == NULL) {
    bottom = msg;
    top = msg;
  } else {
    /* not empty stack */
    msg->previous = top;
    top = msg;
  }
  return 0;
}

int dm510_msgbox_get( char* buffer, int length ) {
  if (top != NULL) {
    msg_t* msg = top;
    int mlength = msg->length;
    top = msg->previous;
```

```
    /* copy message */
    memcpy(buffer, msg->message, mlength);

    /* free memory */
    free(msg->message);
    free(msg);

    return mlength;
  }
  return -1;
}

int main(int argc, char** argv) {
  char *in = "This is a stupid message.";
  char msg[50];
  int msglen;

  /* Send a message containing 'in' */
  dm510_msgbox_put(in, strlen(in)+1);

  /* Read a message */
  msglen = dm510_msgbox_get(msg, 50);

  return 0;
}
```

The following will only be discussed in a later exercises session (week 11):

- Discuss the following source code of a D program. `profile:::tick-1sec` tells the `profile` provider to create a new probe which fires once per second. The function `trace()` indicates that DTrace should record the specific argument and print it out. What are the clauses of the program? What are the actions of the prgram? What are predicates of the program? What happens on execution and what is the output of the program?

```
dtrace:::BEGIN
{
    i = 10;
}

profile:::tick-1sec
/i > 0/
{
    trace(i--);
}

profile:::tick-1sec
/i == 0/
{
    trace("blastoff!");
    exit(0);
}
```