UNIVERSITÄT DUISBURG ESSEN

NES Networked Embedded Systems

Programming in C/C++

Exercises 3

Task 1. Write a small program that overflows the stack. Use repeated printf() so that you can see how big your stack is. Note: The program will crash!
Only on Windows: Use “objdump -p <exefile>” to print the information of the private header. Is there any field that matches (approximately) your result? (Note: All numbers are hexadecimal.)
How can you modify the stack size? (Hint: On Windows it’s a linker option and on Linux a system command.) Change the stack size to 1MB and rerun. What’s the result?
Include all answers in your report!
(code 15pts, report 10pts, comments 5pts)

Task 2: Write a program that creates, deletes and tests dynamic two-dimensional matrix on the heap. The access to each cell of this dynamic matrix should be possible with “matrix[row][col]”! Add a small report as header comment describing how you organize the matrix in memory.

i. The function “`int ** createMatrix(int row, int col)`” should create a dynamic matrix and return it when successful, NULL otherwise. Try to minimize the number of malloc() calls.
ii. The function “`void deleteMatrix(int **matrix)`” should free a dynamic matrix.
iii. The main program gets the number of rows and columns as command line parameter and checks them. After creating the dynamic matrix, it assigns each cell with the value computed from “`row_number + column_number`”. Then it prints out the complete matrix and finally frees it.

Hint: Look at slide 22 (Programming in C) what matrix[row][col] means when it is defined as int **matrix.
(code 55pts, comments 10pts, report 5pts)

Bonus task: **Warning:** You may crash the program or even more if you do it wrong!
Given is the following program fragment (ex3bonus.c on moddle):

```
#include <stdio.h>

void modify(void) {
  /* insert code here */
}

int main(void) {
  int i = 42;
  modify();
  i = 0;
  printf("The answer is %d!\n", i);
}
```

Of course, the output of the program is completely wrong :-) Replace the body of function modify() with code that modifies the return address of the function in such a way that the

“i = 0;” instruction is skipped. Your solution does not need to be generic but can be tailored to this single program and your platform and compiler. Describe how you figure out the right numbers.

Hint: Look again at slide 84 how the stack is organized when a function is called. Make a drawing how you want to access the return address.

Hint 2: Depending on your development environment it might be necessary to switch of all compilation optimizations (“-O0”) since the compiler might optimize away the code you write in modify(). If you compile from command line no optimization is the standard for gcc.

Note: This is a bad hack and in no way good programming practice! It’s to demonstrate how you can change the program flow by purpose or by accident when pointers are used improperly or when buffer overflows occur…
(code 10pts, report 10pts)