OS'17: Assignment 2

Contents

[Important Instructions 1](#_Toc476192881)

[Delivery Method & Dates 1](#_Toc476192882)

[Questions 2](#_Toc476192883)

[1) Create Named Integer Array 2](#_Toc476192884)

[2) Get the Size of the Array 2](#_Toc476192885)

[3) Set Element in the Array 3](#_Toc476192886)

[4) Calculate Array Variance 4](#_Toc476192887)

[BONUS 4](#_Toc476192888)

[Expand a named integer array 4](#_Toc476192889)

[CHALLENGE!!! 7](#_Toc476192890)

[Change the FOS Virtual Space 7](#_Toc476192891)

[

## Important Instructions

Please read instructions carefully, any mistake or error may result in assignment rejection from the automated grading system:

1. Make sure to start solving assignment2 using the **assignment2.startup** code from:[**https://www.dropbox.com/sh/g28609in5q78cml/AACVQt1WtJegj7zDg8neojXJa?dl=0**](https://www.dropbox.com/sh/g28609in5q78cml/AACVQt1WtJegj7zDg8neojXJa?dl=0)

Make a copy of your previous code, then copy and replace the five files from the above link into your "**/kern**". Finally, if you want, merge back your code.

1. During your solution, make sure that you **DON'T CHANGE** any other file rather than command\_prompt.c/.h
2. Your code *MUST* be written inside the given function for each question (as specified [below](#_Questions)).

ANY violation to this rule, or change of function names, or deletion of template functions may result in assignment rejection by the automated grading system.

1. Assignments *MUST* be delivered using this online sheet: <https://docs.google.com/forms/d/e/1FAIpQLScu92hBt-mc_whoYsC17J1i8o0EQFYznM3g6sKribSC2icNrA/viewform>
2. If you face a problem in **RE-submitting** the assignment code, kindly follow [these steps](https://www.dropbox.com/s/rqoziu62r1s270z/Resubmission%20Issue%20Solution.txt?dl=0) [Resubmission ONLY]

## Delivery Method & Dates

* **Assignment & Bonus:** through the above online form
  + **Early delivery (main questions):** THU of 4th week (9-3-2017 23:59)
  + **Final delivery (main questions + bonus):** THU of 5th week (16-3-2017 23:59)
* **Challenge:** during the office hours
  + **due to** THU of 6th week (23-3-2017)

## Questions

Add the following commands to the Kernel:

### Create Named Integer Array

#### Name:

cnia <array name> <size>

#### Description:

* This command should create an integer array with the given name and size, and initialize it by **ZEROs**.
* It should return the **start virtual address of the FIRST ELEMENT** in allocated array.
* The allocation should start initially from the virtual address **0xF1000000**, and then **continue allocating** after this address.
* MAX number of arrays to be created is 30.

#### Example:

**FOS>** cnia x 3 //allocate array "x" of size 3 and initialize it by ZEROs, starting from **0xF100000**)

**FOS>** cnia myArr 4 //allocate array "myArr" of size 4 after the array "x" and initialize it by ZEROs

#### Function:

Your code MUST be written inside the following function:

**int\* CreateIntArray(char\*\* arguments)**

* + arguments[1]: array name
  + arguments[2]: array size
  + **Return:** start address of the **FIRST ELEMENT** in the created array

#### Helper Functions:

* strcpy(char \*dst, const char \*src): to copy string from src to dst

### Get the Size of the Array

#### Name:

gas <array name>

#### Description:

This command should return the size of a previously created array with the given <array name>.

If the array is exists, return its size

Else, return -1

#### Example:

**FOS>** cnia x 3

**FOS>** cnia myArr 14

**FOS>** cnia zeros 10

**FOS>** gas x //should return 3

**FOS>** gas myArr //should return 14

**FOS>** gas zero //should return -1

**FOS>** gas zeros //should return 10

#### Function:

Your code MUST be written inside the following function:

**int GetArraySize(char\*\* arguments)**

* + arguments[1]: array name

**Return:**

* + If the array is exist, return its size
  + Else: return -1

#### Helper Functions:

* strcmp(const char \*p, const char \*q): to compare string p to string q

### Set Element in the Array

#### Name:

sea <array name> <zero-based index> <value>

#### Description:

This command should set the element at <zero-based index> inside the given array <array name> by the given <value>.

If the index is correct (i.e. within array bounds), set the element and return

else, do nothing and return -1.

#### Example:

**FOS>** cnia x 3

**FOS>** cnia myArr 4

**FOS>** sea x 2 300 //should set the 3rd element in array “x” by 300

**FOS>** sea myArr 0 100000 //should set the 1st element in array “myArr” by 100000

**FOS>** sea x 3 11 //should do nothing and return -1

#### Function:

Your code MUST be written inside the following function:

**int SetElementInArray(char\*\* arguments)**

* + arguments[1]: array name
  + arguments[2]: zero-based index
  + arguments[3]: value to be set

**Return**

* + If index is correct (within array bounds): return 0
  + Else: return -1

#### Helper Functions:

* strcmp(const char \*p, const char \*q): to compare string p to string q

### Calculate Array Variance

#### Name:

cav <array name>

#### Description:

* This command should calculate the variance of the elements in the given <array name>, according to the following equation:

Where:

* N: array size
* : is the mean (average) of the array elements:
* **NOTE:** use **integer** data types (no float, no double)

#### Example:

**FOS>** cnia x 3

**FOS>** cnia y 4

**FOS>** sea x 0 10

**FOS>** sea x 1 20

**FOS>** sea x 2 30

**FOS>** cav x //should print 66

**FOS>** sea y 0 400

**FOS>** sea y 1 400

**FOS>** cav y //should print 40,000

#### Function:

Your code MUST be written inside the following function:

**int CalcArrVar(char\*\* arguments)**

* + arguments[1]: array name

#### Helper Functions:

* strcmp(const char \*p, const char \*q): to compare string p to string q

## BONUS

Expand a named integer array

#### Name:

enia <array name> <num of extra elements>

#### Description:

This command should expand the previously created <array name> by the given <num of extra elements> and **initialize** them by **ZEROs**. This is done by moving up all allocated arrays that are located after the deleted one.

**Note:** you should **update the pointers of all moved arrays** after expansion.

#### Ex:

**FOS>** cnia x 2 //allocate array "x" of size 2 starting from **0xF1000000**

**FOS>** cnia y 3 //allocate array "y" of size 3 after the array "x"

**FOS>** cnia z 1 //allocate array "z" of size 1 after the array "y"

The above commands should allocate three arrays behind each other starting from address **0xF1000000**, as shown in figure 1.

...

...

4 GB

z

0xF10000**00**

0

y

x

Start of y

Start of z

**Virtual Space**

Figure : Virtual space layout after allocating three different arrays

**FOS>** sea y 0 400 // y[0] = 400

**FOS>** sea y 2 400 // y[2] = 400

**FOS>** sea z 0 400 // z[0] = 400

**FOS>** enia x 2 //expand the array "x" by 2 extra elements and move up all the above arrays

The above command should expand the array "x" by 2 elements and move up all the two arrays after it together with updating their corresponding pointers, as shown in figure 2.

...

...

4 GB

z

0xF10000**00**

0

y

x

NEW start of y

NEW start of z

**Virtual Space**

Figure : Virtual space layout after expanding "x"

**FOS>** enia z 3 //expand the array "z" by 3 extra elements

**FOS>** enia y 1 //expand the array "y" by 1 extra element and move up the above array

**FOS>** sea x 2 400 // x[2] = 400

**FOS>** sea x 3 400 // x[3] = 400

**FOS>** sea z 3 400 // z[3] = 400

**FOS>** cnia w 4 //allocate array "w" of size 4 after the array "z"

**FOS>** sea w 0 200 // w[0] = 200

**FOS>** sea w 3 200 // w[3] = 200

**FOS>** gas x //should print 4

**FOS>** gas y //should print 4

**FOS>** gas z //should print 4

**FOS>** gas w //should print 4

**FOS>** cav x //should print 40,000

**FOS>** cav y //should print 40,000

**FOS>** cav z //should print 40,000

**FOS>** cav w //should print 10,000

#### Function:

Your code MUST be written inside the following function:

**void ExpandIntArray(char\*\* arguments)**

* + arguments[1]: array name

## CHALLENGE!!!

Change the FOS Virtual Space

Change the start location of the FOS virtual space to be @ address 0 instead of KERNEL\_BASE

After this, address *N* in the virtual memory should be mapped to address *N* in the physical RAM

**MAKE SURE that you followed the above instructions carefully. Good Luck isA ☺**