A4 Hints

Creating User Level Processes

Implement CallStartUserProcess

- It just calls ProcessManager.StartUserProcess(args)

StartUserProcess

- Get a new process. Set its thread to onCpuThread. Set it to current process. Remember to also link the thread to the process.
- Open the file using fileSystem.Open(INIT_NAME, fileSystem.rootDirectory,O_READ,0)
- Check permission with (fcb.inode.mode & MODE_EXE) == MODE_EXE
- Use LoadExecutable to load address space into the new process addrSpace -> close the file afterwards.
- Set working directory of process using Proc.workingDir = fileSystem.rootDirectory.NewReference()
- Figure out stackTop and sysStackTop
- Start morphing into the user thread
  - Set interrupts to disabled
  - Set onCpuThread to userThread
  - Set the MMU to use the new page table using: addrSpace.SetToThisPageTable()
  - Call BecomeUserThread passing the required parameters.

System Call Handlers Implementation

System Call Handlers Implementation

1. Valid_User_Pointer -- Used in some handle calls
    1. Checks if arguments are invalid (example negative size)
    2. Checks if virtual address is valid (greater than zero and less than max address)
    3. Checks if pages are writable, ONLY if toStore argument is true. Use addrSpace.IsWritable() on every page to make sure they are all writable.
2. Handle_Sys_Fork
3. Handle_Sys_Join
4. Handle_Sys_Exec
5. Handle_Sys_Open
6. Handle_Sys_Read
7. Handle_Sys_Write
8. Handle_Sys_Seek
9. Handle_Sys_Stat
10. Handle_Sys_Chdir

System Call Handlers Implementation

- System call handlers
  - Just print the integers
  - Return the correct code.
  - Print the arguments using the correct method
    - printIntVar
    - printHexVar
    - Print
  - You have to use the following method in address space to get the String from the virtual address. We are not in that virtual address space anymore, so using the pointer will not work:
    - addrSpace.GetStringFromVirtual(&localName, filename asInteger, MAX_STRING_SIZE)

System Call Handlers Implementation

- Handle_Sys_Exec
  - Unlike the rest, we will not simply print the arguments and return in this system call.
  - Handle_Sys_Exec (filename: ptr to array of char, args: ptr to array of ptr to array of char)
  - We want to call processManager.ExecNewProgram() with the same arguments ...
    - Translate the filename using the GetStringFromVirtual method
    - Ensure user pointer is valid for args
    - Call the method (ExecNewProgram()) with the translated filename but use the same arguments.

ExecNewProgram

ExecNewProgram

- Load new program from file system
- Check permissions
- Load a new address space
- Figure out the stacks
- Copy arguments
    - If copy arguments fail, return all the frames of the newly created address space, and then return the method.
- Do what it takes to morph into new program
    - Must throw away old address space
        - After copying the arguments, we don’t’ need the old address space anymore. Return all its frames.
    - Set current process address space to the new address space.
    - Restart this thread as a "new" user thread
        - Disable interrupt
        - Set Page table
        - Call BecomeUserThread passing the required parameters.

Error Codes

- Check Syscall.h for the error codes.

Command Line Arguments

- Inside ExecNewProgram() use a helper method to copy the arguments to the new address space.
- How to do the copying?
  - Use CopyBytesFromVirtual and CopyBytesToVirtual in AddrSpace objects.
  - The trick, is to figure out what address to read from and write to.

Command Line Arguments

1. Check and ensure the argument pointer is valid.
2. Get number of arguments
   - This is a pointer to an array (of points to array of chars)
   - In KPL, arrays carry their size with them.
   - The first 4 bytes in an array address carry the number of elements in an array.

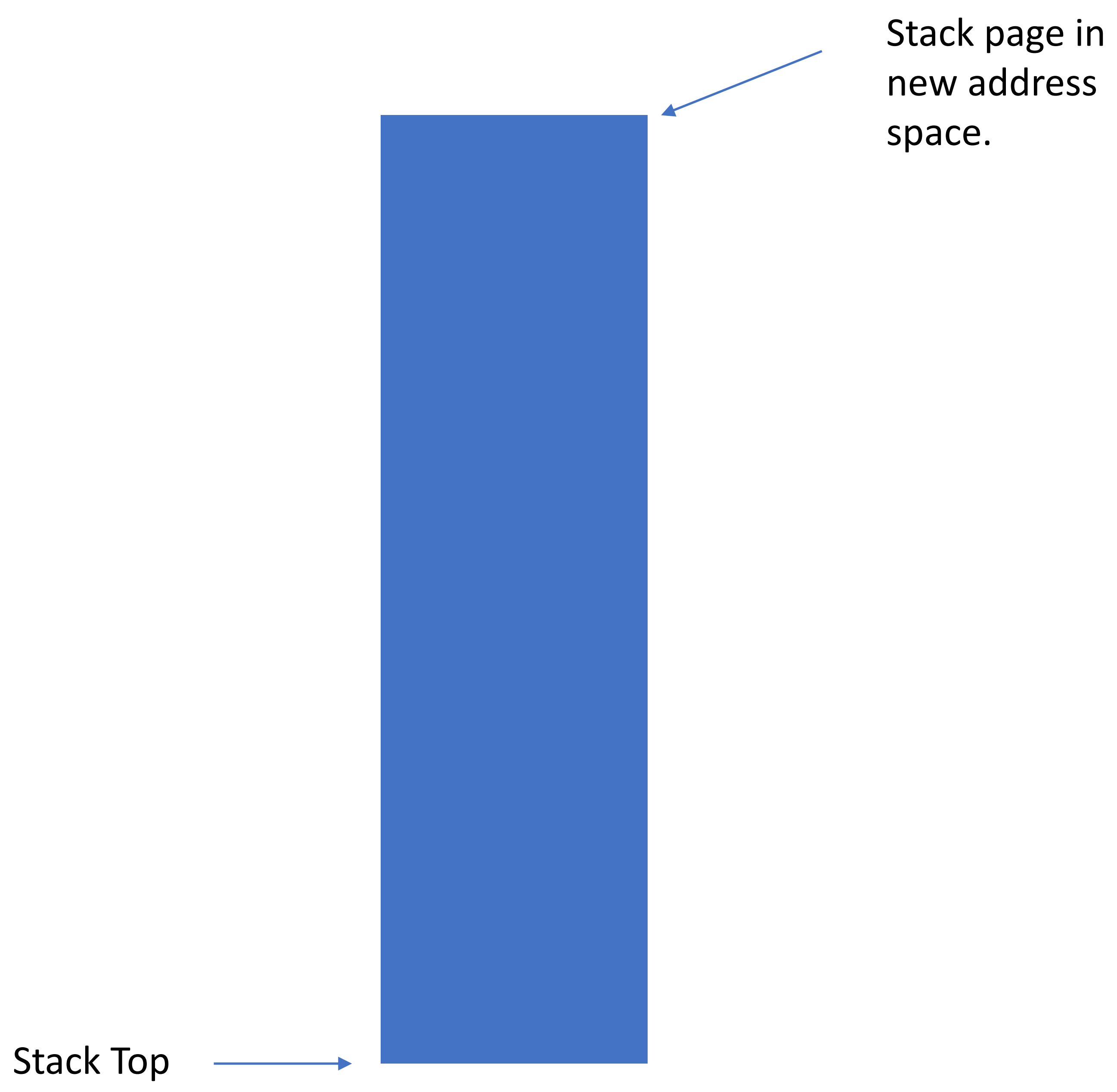
   ```
   int numArgs
   AddrSpace.CopyBytesFromVirtual ((&numArgs) asInteger, argsPtr, 4)
   ```
3. Calculate new stack top by decreasing it according to how many bytes you need to store the array. Remember, we're storing the arguments on the stack, that's why we need to change the new stack top value.

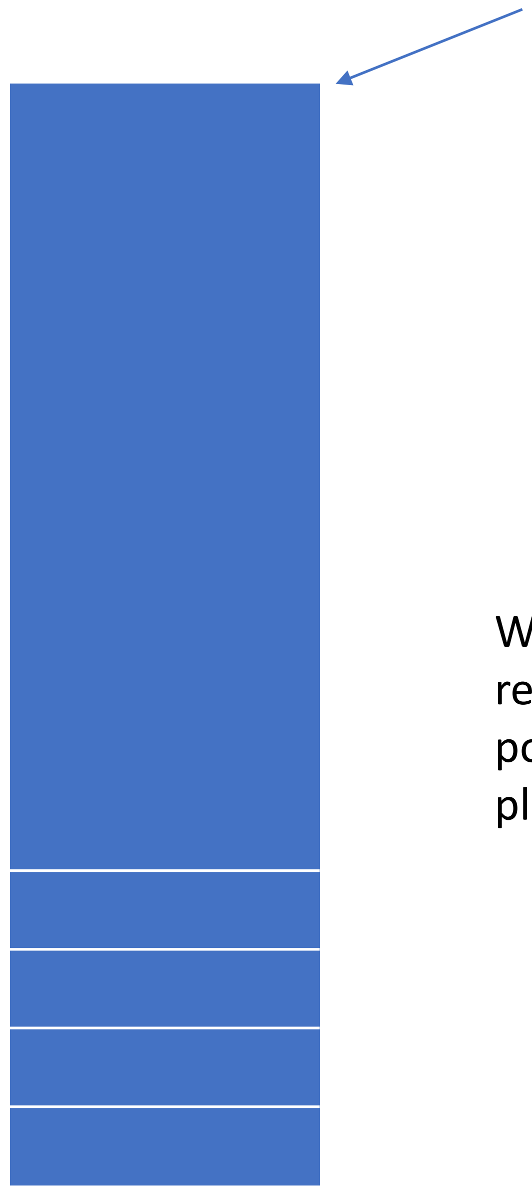
Command Line Arguments

- For every array:
    1. Keep track of where you need to store the pointer to that array (it should be in some of the space you calculated in the initial array)
    2. Figure out its size (same way but watch out: array of char are 4 bytes for size + 1 byte for each char. You still have to allocate in multiples of 4 so round up to a multiple of 4 after you do the calculation)
    3. Figure out the virtual address based of the new bytes, by moving the stack pointer further.
    4. Write the bytes from the old address space to the new address.
    5. Write the address of the first byte to where the pointer need to be stored (step 1)

Command Line Arguments

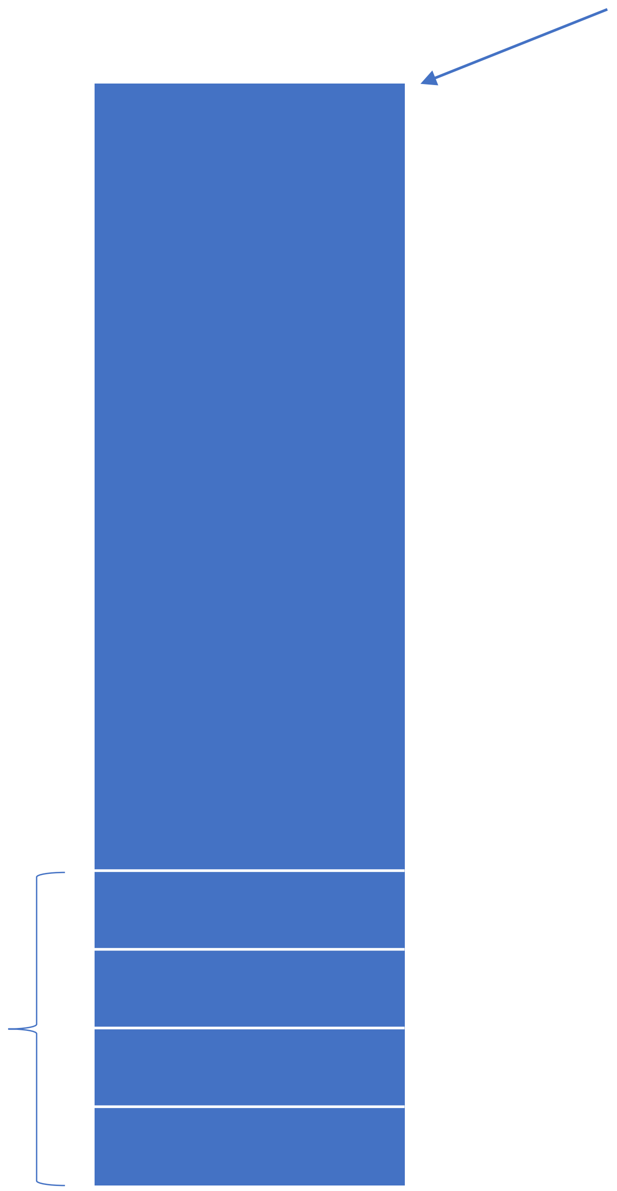
- CopyBytesFromVirtual
  - Copies from a virtual address space to physical (kernel) memory.
- GetStringFromVirtual
  - Same: copies from virtual to physical.
- There is no way to copy from old virtual to new virtual.
  - We have to figure out the physical address of the of where to store the array of char.
  - Use method ExtractFrameAddr to extract the last page in the new Addr space.
  - Now you got the physical address of the new stack page. You still have not figured out the exact physical address.
    - Add to it the (new virtual address) % PAGE_SIZE

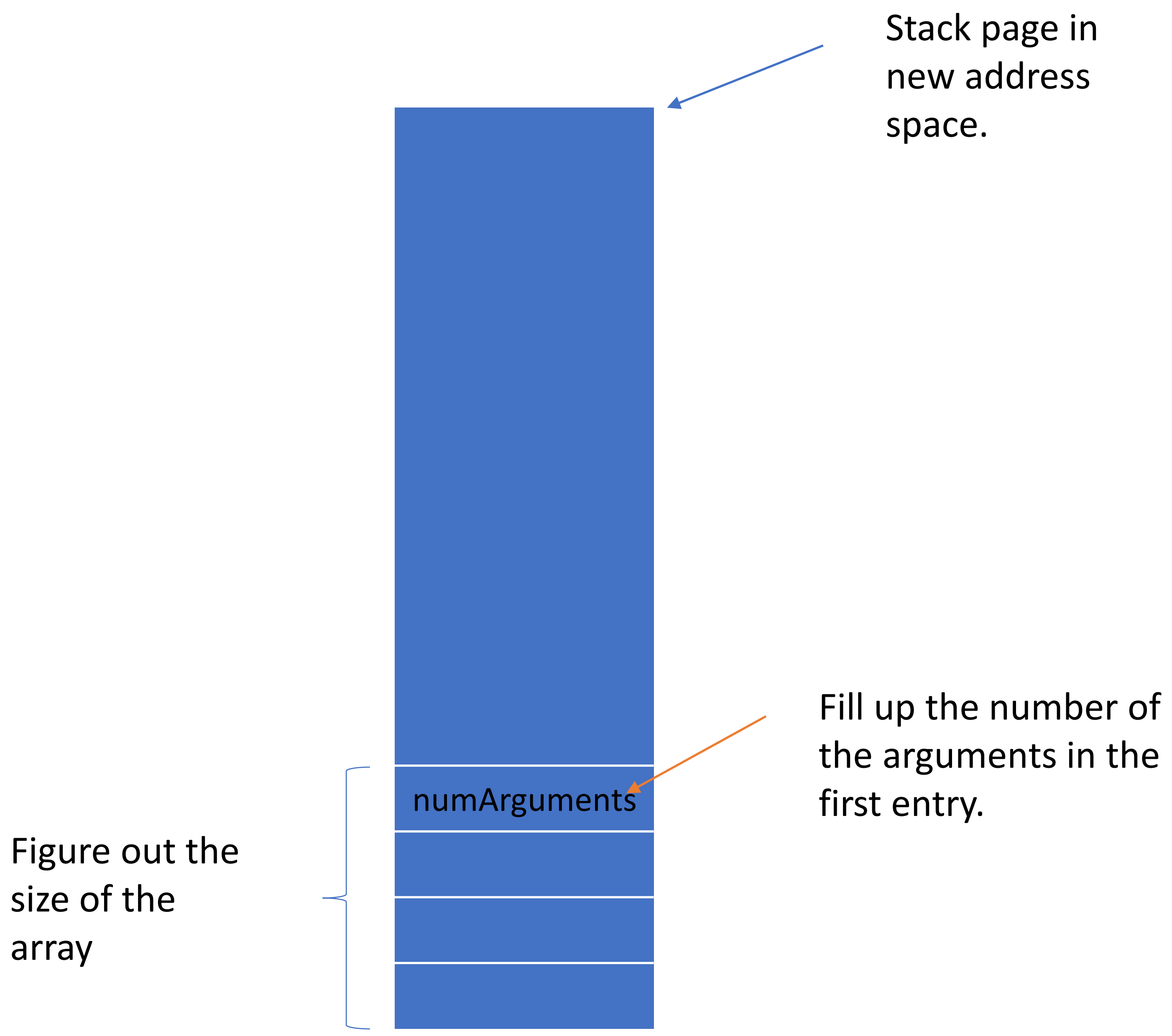

Stack page in new address space.
Stack Top

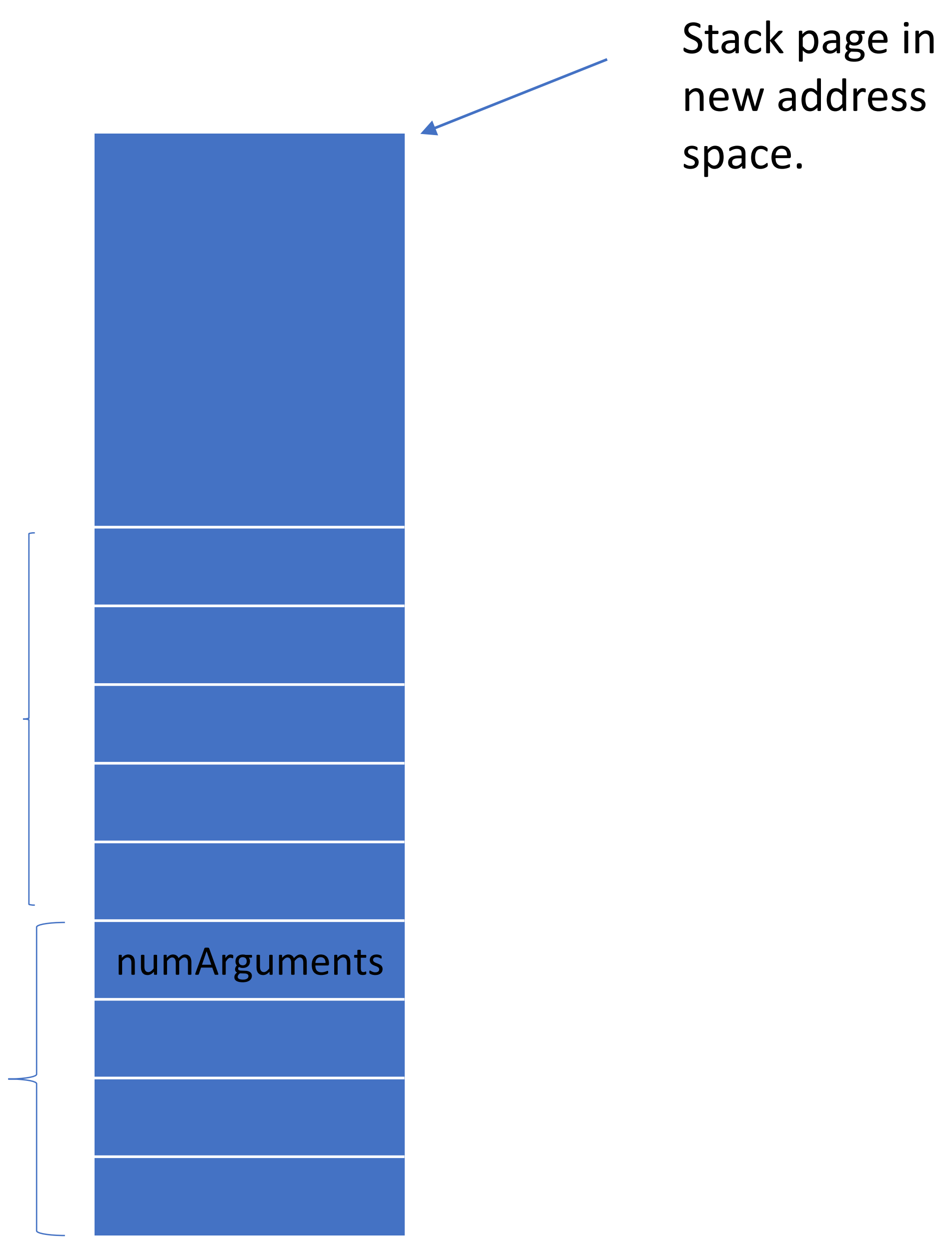


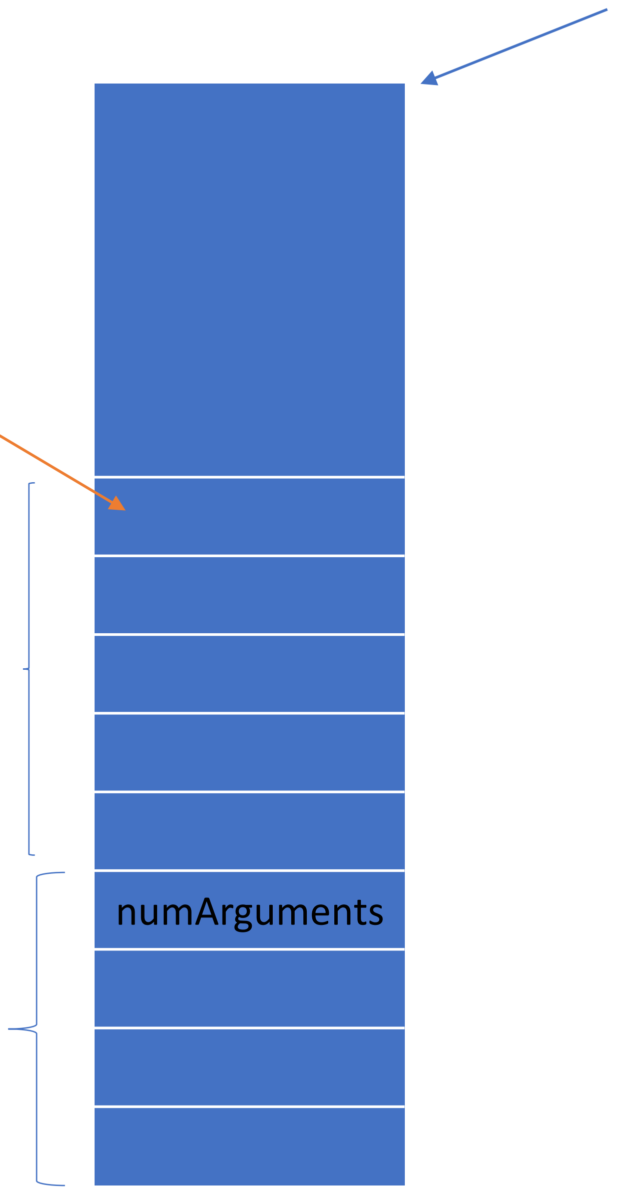
Stack page in new address space.

We have to manually recreate the array of pointers to char and place it in top stack.


Stack page in new address space.
Figure out the size of the array


Stack page in new address space.
Fill up the number of the arguments in the first entry.
numArguments
Figure out the size of the array


Stack page in new address space.
Figure out the size of first char array
numArguments
Figure out the size of the array

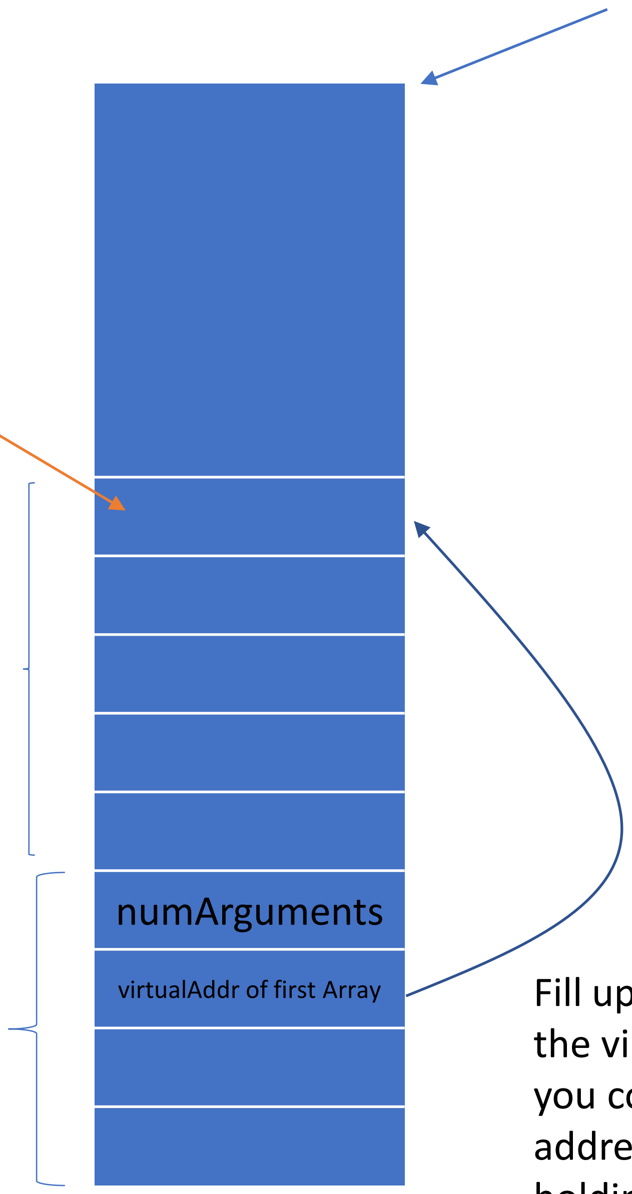

Stack page in new address space.
Copy first array of char from old virtual address. Start with copying the number of arguments and then the elements.
Figure out the size of first char array
numArguments
Figure out the size of the array

Stack page in new address space.

Copy first array of char from old virtual address. Start with copying the number of arguments and then the elements.

Figure out the size of first char array

Figure out the size of the array



Fill up the args array with the virtual address of arrays you copied. The virtual address is the address holding the size of the array.