Sharif-OS-Lab /
summer1403-6-99101087_99100422

Code · Issues 1 · Pull requests · Actions · Projects · Security · Insights

Edit · New issue · Jump to bottom

# Session 6 Report #1

Open · 36 tasks done · BozorgmehrZia opened this issue on Aug 4 · 0 comments

Assignees

Labels documentation

BozorgmehrZia commented on Aug 4 • edited

Team Name: `99101087-99100422`

Student Name of member 1: `AmirReza Azari`
Student No. of member 1: `99101087`

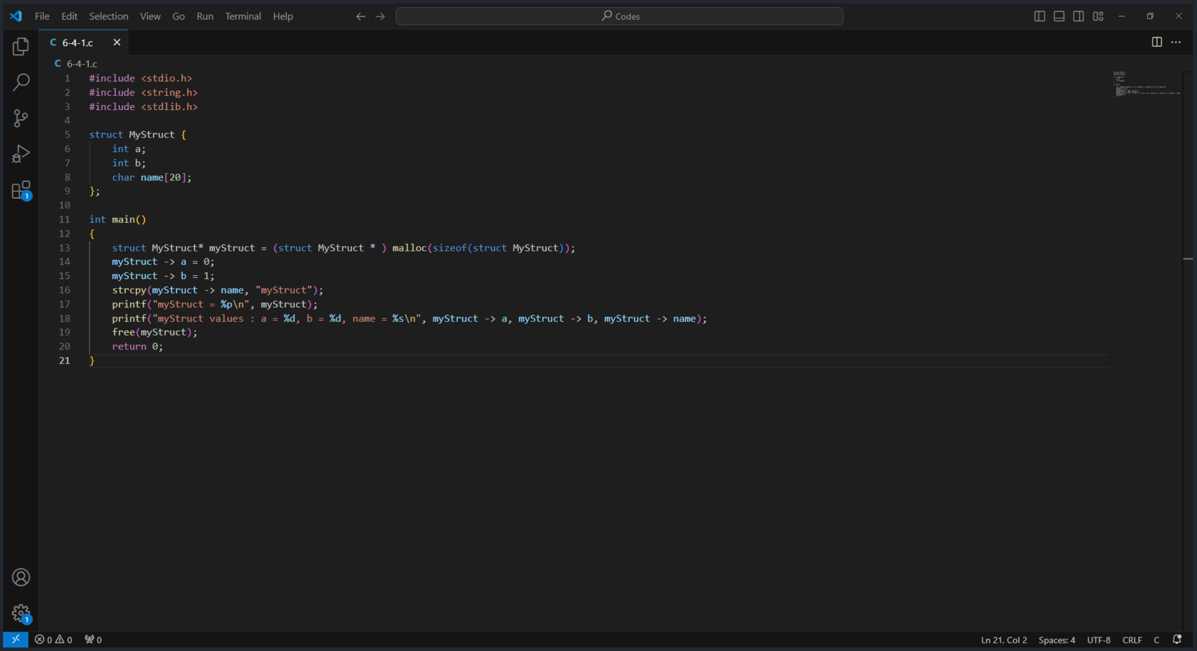
Student Name of member 2: `Bozorgmehr Zia`
Student No. of member 2: `99100422`

- ☑ Read Session Contents.

## Section 6.4

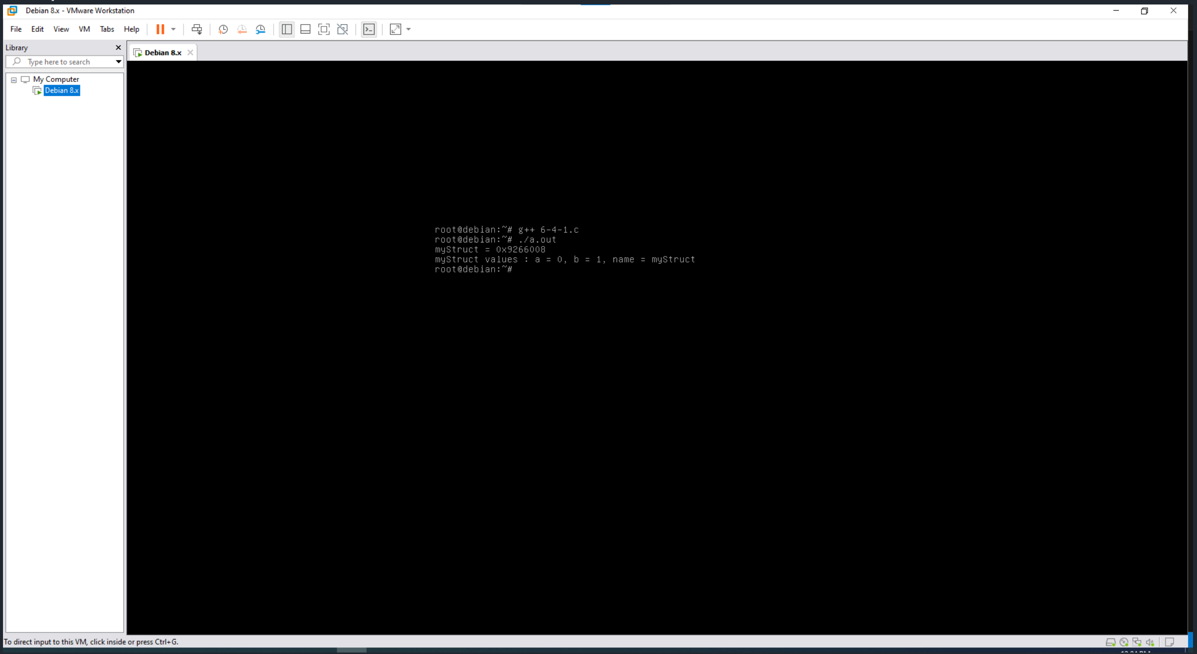
- ☑ Using `malloc` and `free` in program

☑ Code (in file 6-4-1.c):



```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

struct MyStruct {
    int a;
    int b;
    char name[20];
};

int main()
{
    struct MyStruct* myStruct = (struct MyStruct * ) malloc(sizeof(struct MyStruct));
    myStruct -> a = 0;
    myStruct -> b = 1;
    strcpy(myStruct -> name, "myStruct");
    printf("myStruct = %p\n", myStruct);
    printf("myStruct values : a = %d, b = %d, name = %s\n", myStruct -> a, myStruct -> b, myStruct -> name);
    free(myStruct);
    return 0;
}
```

☑ Output:

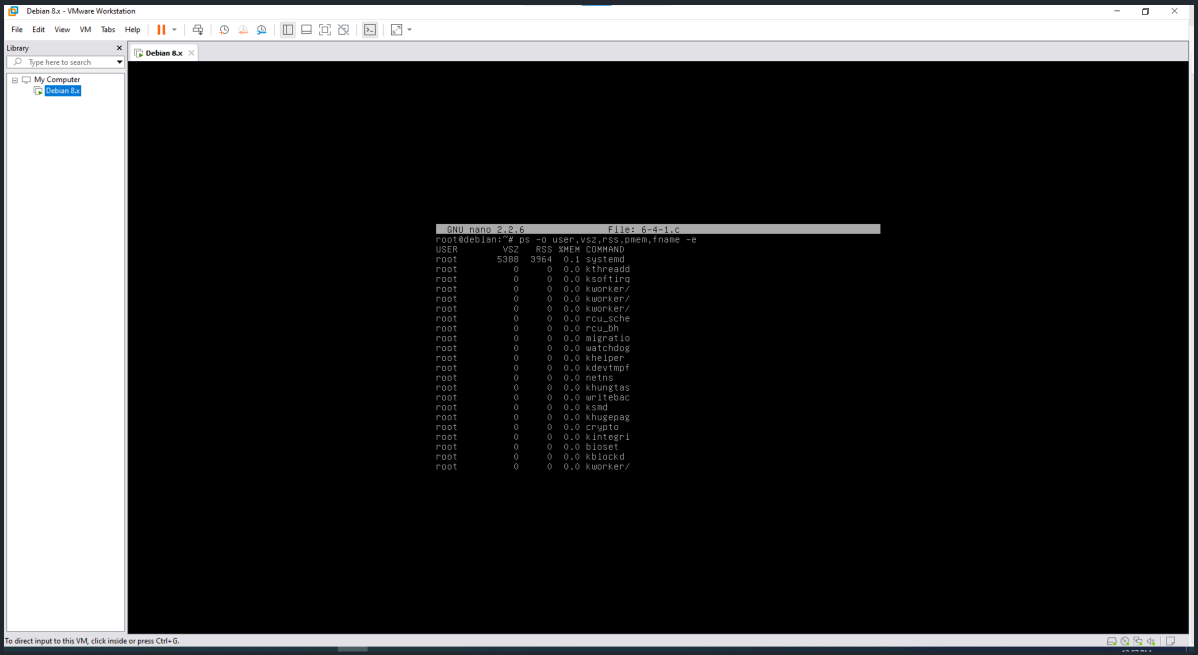

Debian 8.x - VMware Workstation
File
Edit
View
VM
Tabs
Help
Library
Type here to search
My Computer
Debian 8.x
Debian 8.x
To direct input to this VM, click inside or press Ctrl+G.

```
root@debian:~# g++ 6-4-1.c
root@debian:~# ./a.out
myStruct = 0x9266008
myStruct values : a = 0, b = 1, name = myStruct
root@debian:~#
```

The `malloc` function returns a void pointer to the reserved space. The return value is NULL if not enough storage is available, or if size was specified as zero.

☑ Using `ps`

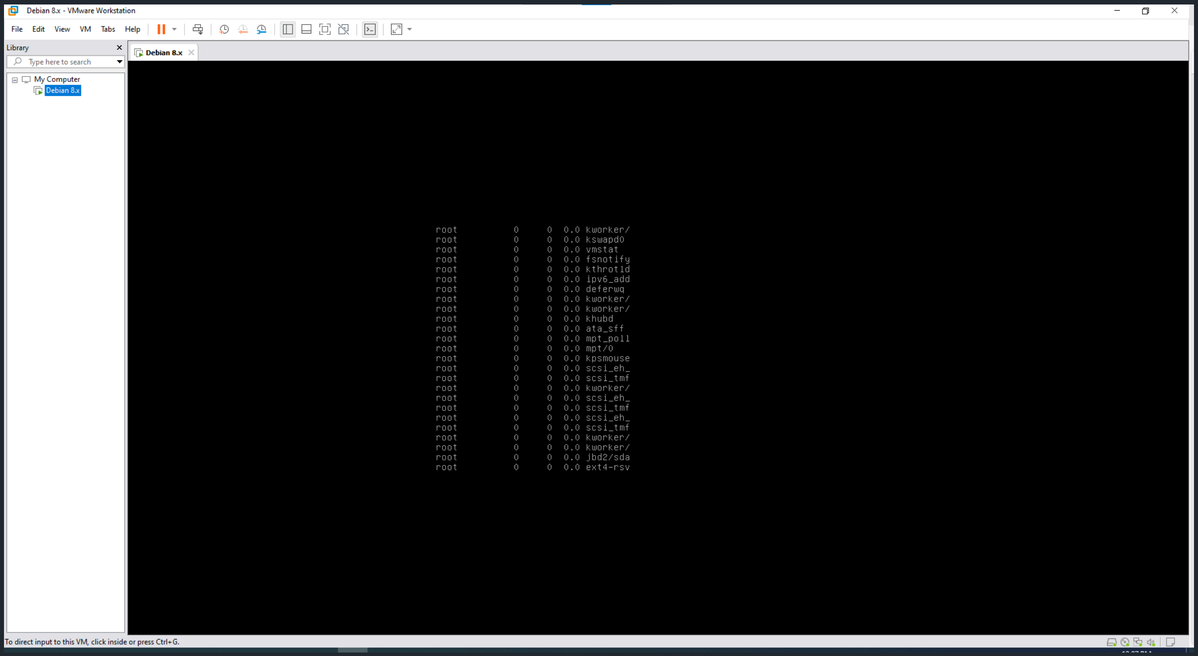
☑



```
  GNU nano 2.2.6               File: 6-4-1.c
root@debian:~# ps -o user,vsz,rss,pmem,fname -e
USER         VSZ   RSS %MEM COMMAND
root        5388  3964  0.1 systemd
root           0     0  0.0 kthreadd
root           0     0  0.0 ksoftirq
root           0     0  0.0 kworker/
root           0     0  0.0 kworker/
root           0     0  0.0 kworker/
root           0     0  0.0 rcu_sche
root           0     0  0.0 rcu_bh
root           0     0  0.0 migratio
root           0     0  0.0 watchdog
root           0     0  0.0 khelper
root           0     0  0.0 kdevtmpf
root           0     0  0.0 netns
root           0     0  0.0 khungtas
root           0     0  0.0 writebac
root           0     0  0.0 ksmd
root           0     0  0.0 khugepag
root           0     0  0.0 crypto
root           0     0  0.0 kintegri
root           0     0  0.0 bioset
root           0     0  0.0 kblockd
root           0     0  0.0 kworker/
```

To direct input to this VM, click inside or press Ctrl+G.

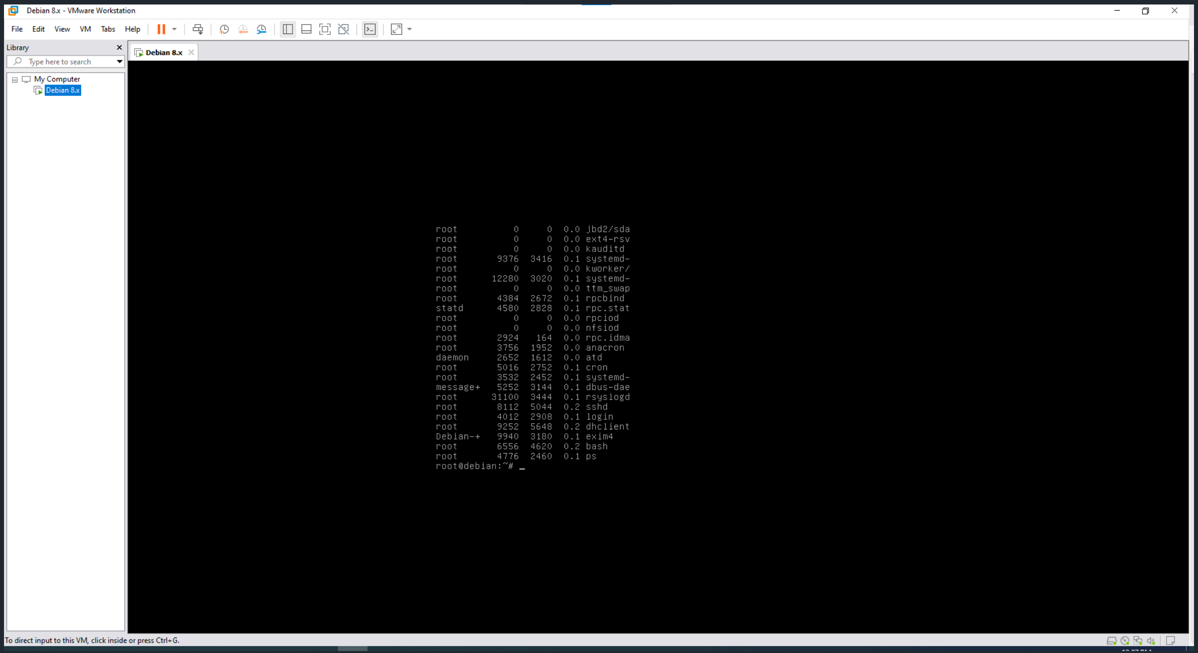
☑



```
root           0     0  0.0 kworker/
root           0     0  0.0 kswapd0
root           0     0  0.0 vmstat
root           0     0  0.0 fsnotify
root           0     0  0.0 kthrotld
root           0     0  0.0 ipv6_add
root           0     0  0.0 deferwq
root           0     0  0.0 kworker/
root           0     0  0.0 kworker/
root           0     0  0.0 khubd
root           0     0  0.0 ata_sff
root           0     0  0.0 mpt_poll
root           0     0  0.0 mpt/0
root           0     0  0.0 kpsmouse
root           0     0  0.0 scsi_eh_
root           0     0  0.0 scsi_tmf
root           0     0  0.0 kworker/
root           0     0  0.0 scsi_eh_
root           0     0  0.0 scsi_tmf
root           0     0  0.0 scsi_eh_
root           0     0  0.0 scsi_tmf
root           0     0  0.0 kworker/
root           0     0  0.0 kworker/
root           0     0  0.0 jbd2/sda
root           0     0  0.0 ext4-rsv
```

To direct input to this VM, click inside or press Ctrl+G.

- ☑



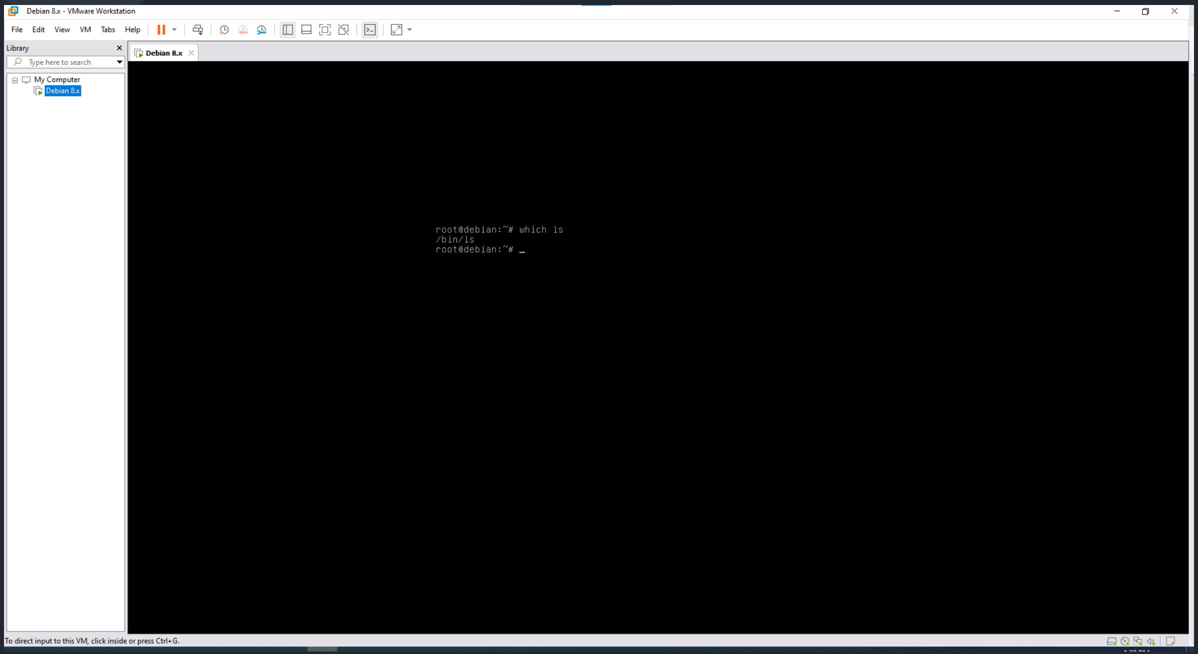
Debian 8.x - VMware Workstation
File Edit View VM Tabs Help
Library
Type here to search
My Computer
Debian 8.x
Debian 8.x
To direct input to this VM, click inside or press Ctrl+G.

```
root            0      0  0.0 jbd2/sda
root            0      0  0.0 ext4-rsv
root            0      0  0.0 kauditd
root         9376   3416  0.1 systemd-
root            0      0  0.0 kworker/
root        12280   3020  0.1 systemd-
root            0      0  0.0 ttm_swap
root         4384   2672  0.1 rpcbind
statd        4580   2828  0.1 rpc.stat
root            0      0  0.0 rpciod
root            0      0  0.0 nfsiod
root         2924    164  0.0 rpc.idma
root         3756   1952  0.0 anacron
daemon       2652   1612  0.0 atd
root         5016   2752  0.1 cron
root         3532   2452  0.1 systemd-
message+     5252   3144  0.1 dbus-dae
root        31100   3444  0.1 rsyslogd
root         8112   5044  0.2 sshd
root         4012   2908  0.1 login
root         9252   5648  0.2 dhclient
Debian-+     9940   3180  0.1 exim4
root         6556   4620  0.2 bash
root         4776   2460  0.1 ps
root@debian:~# _
```

- ☑ Columns are:
  - ☑ `user` : This column displays the user ID or username of the person who owns and runs the process. Example Output: root, user1. Alias: uname.
  - ☑ `vsz` : This column shows the virtual memory size of the process in kilobytes. It's the total memory size that a process can access, such as the memory of the shared libraries, the allocated memory such as stack and heap, and swapped out memory.
    Example Output: 1048576 (for 1 GB). Alias: vsize.
  - ☑ `rss` : This column shows the resident set size, which is the non-swapped physical memory a task has used (in kilobytes). It doesn't show all of the information about the memory, like loaded libraries, heap, or stack. It shows the shared libraries if their pages are in memory.
    Example Output: 2048 (for 2 MB).
  - ☑ `pmem` : This column shows the percentage of resident set size (RSS) to the physical memory used by the process. Example Output: 0.1
  - ☑ `fname` : This column displays the first 8 bytes of the process's executable file name. The output in this column may contain spaces. Example Output: bash, python

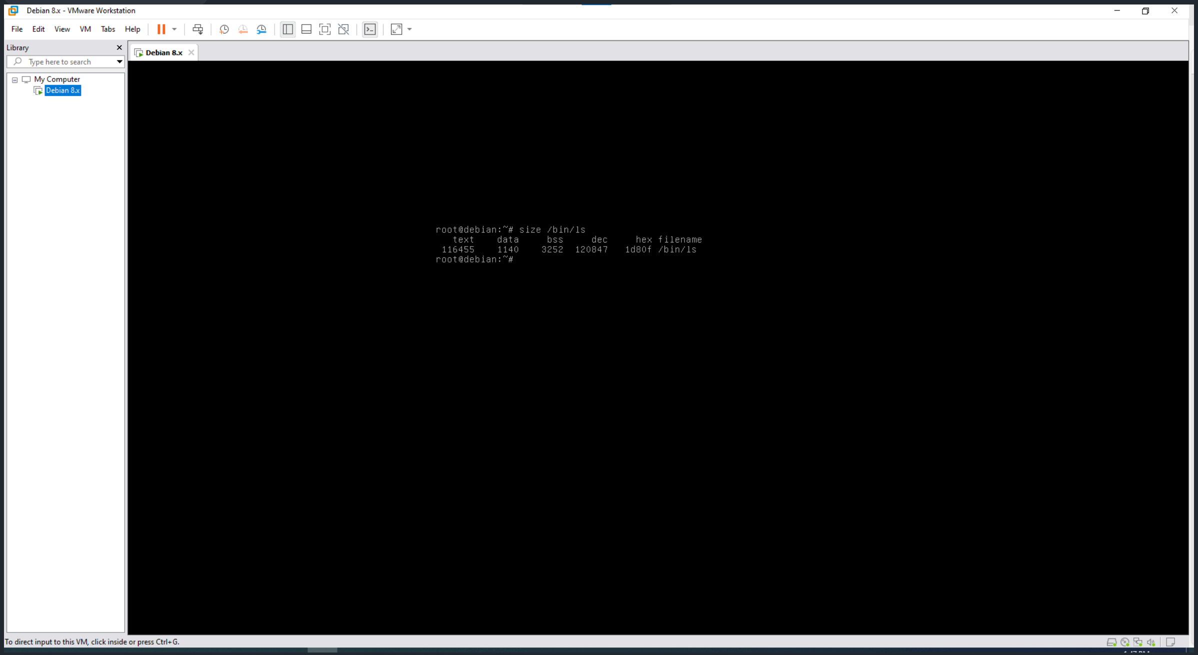
- ☑ Getting started with memory segments

☑ `which ls`


Debian 8.x - VMware Workstation
File Edit View VM Tabs Help
Library
Type here to search
My Computer
Debian 8.x
Debian 8.x
To direct input to this VM, click inside or press Ctrl+G.

```
root@debian:~# which ls
/bin/ls
root@debian:~# _
```

☑ `size /bin/ls`


Debian 8.x - VMware Workstation
File Edit View VM Tabs Help
Library
Type here to search
My Computer
Debian 8.x
Debian 8.x
To direct input to this VM, click inside or press Ctrl+G.

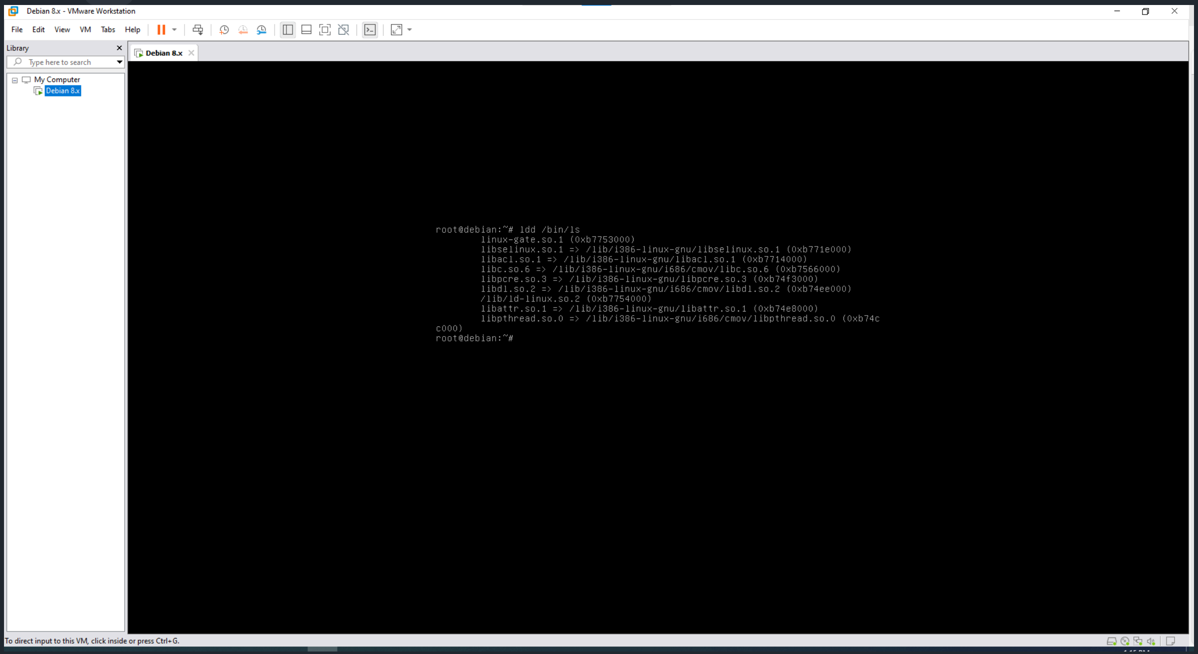
```
root@debian:~# size /bin/ls
   text    data     bss     dec     hex filename
 116455    1140    3252  120847   1d80f /bin/ls
root@debian:~#
```

As you see, 116455 bytes from memory are allocated to text part, 1140 bytes for initialized data, 3252 bytes for bss, and 120847 bytes for dec.

☑ This command doesn't show stack and heap size.

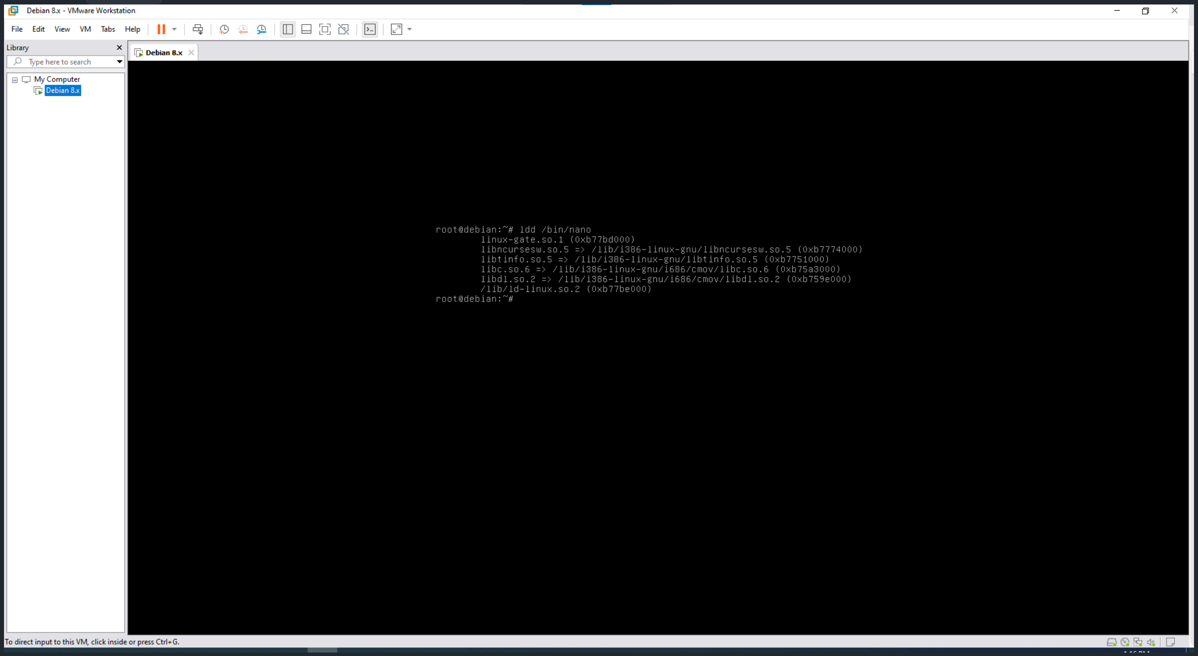
☑ Getting started with memory sharing

☑ lld for `ls`


Debian 8.x - VMware Workstation
File
Edit
View
VM
Tabs
Help
Library
Type here to search
My Computer
Debian 8.x
Debian 8.x
To direct input to this VM, click inside or press Ctrl+G.

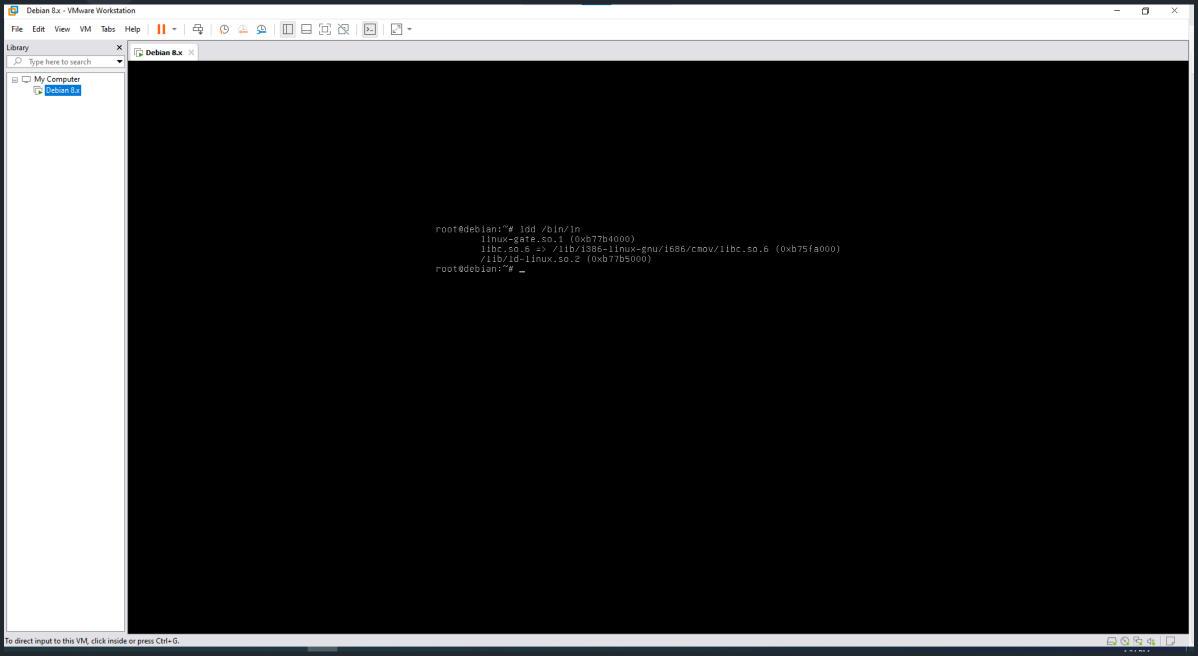
```
root@debian:~# ldd /bin/ls
        linux-gate.so.1 (0xb7753000)
        libselinux.so.1 => /lib/i386-linux-gnu/libselinux.so.1 (0xb771e000)
        libacl.so.1 => /lib/i386-linux-gnu/libacl.so.1 (0xb7714000)
        libc.so.6 => /lib/i386-linux-gnu/i686/cmov/libc.so.6 (0xb7566000)
        libpcre.so.3 => /lib/i386-linux-gnu/libpcre.so.3 (0xb74f3000)
        libdl.so.2 => /lib/i386-linux-gnu/i686/cmov/libdl.so.2 (0xb74ee000)
        /lib/ld-linux.so.2 (0xb7754000)
        libattr.so.1 => /lib/i386-linux-gnu/libattr.so.1 (0xb74e8000)
        libpthread.so.0 => /lib/i386-linux-gnu/i686/cmov/libpthread.so.0 (0xb74c
c000)
root@debian:~#
```

☑ lld for `nano`


Debian 8.x - VMware Workstation
File
Edit
View
VM
Tabs
Help
Library
Type here to search
My Computer
Debian 8.x
Debian 8.x
To direct input to this VM, click inside or press Ctrl+G.

```
root@debian:~# ldd /bin/nano
        linux-gate.so.1 (0xb77bd000)
        libncursesw.so.5 => /lib/i386-linux-gnu/libncursesw.so.5 (0xb7774000)
        libtinfo.so.5 => /lib/i386-linux-gnu/libtinfo.so.5 (0xb7751000)
        libc.so.6 => /lib/i386-linux-gnu/i686/cmov/libc.so.6 (0xb75a3000)
        libdl.so.2 => /lib/i386-linux-gnu/i686/cmov/libdl.so.2 (0xb759e000)
        /lib/ld-linux.so.2 (0xb77be000)
root@debian:~#
```

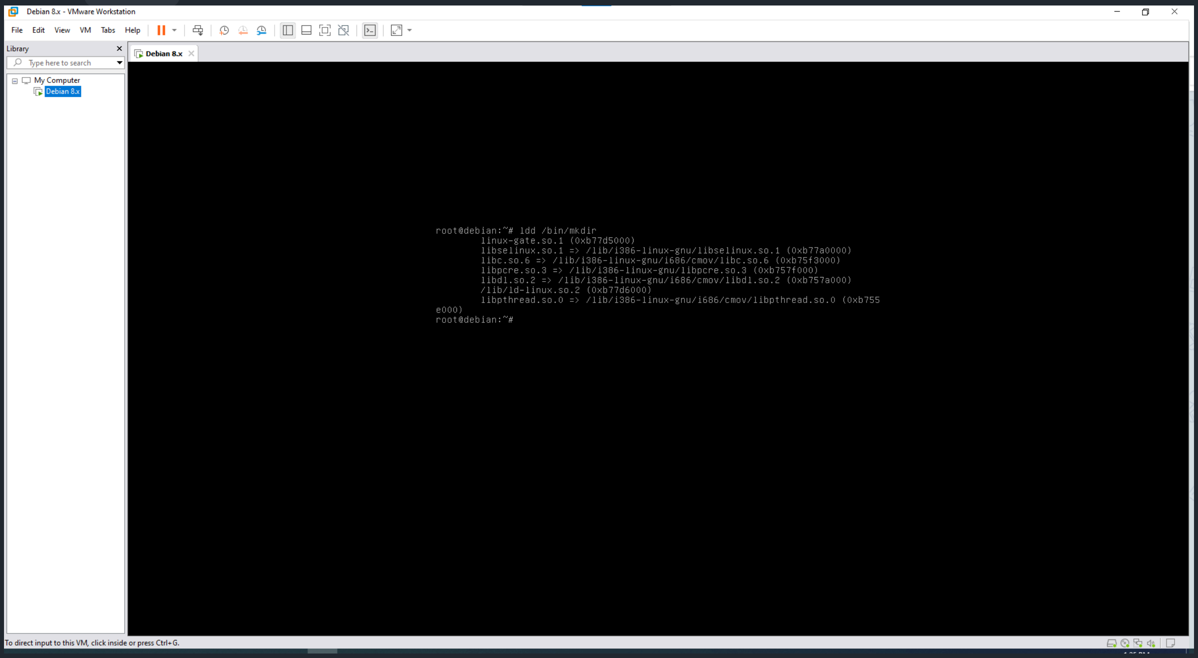
- ☑ lld for `ln`



```
root@debian:~# ldd /bin/ln
        linux-gate.so.1 (0xb77b4000)
        libc.so.6 => /lib/i386-linux-gnu/i686/cmov/libc.so.6 (0xb75fa000)
        /lib/ld-linux.so.2 (0xb77b5000)
root@debian:~# _
```

To direct input to this VM, click inside or press Ctrl+G.

- ☑ lld for `mkdir`

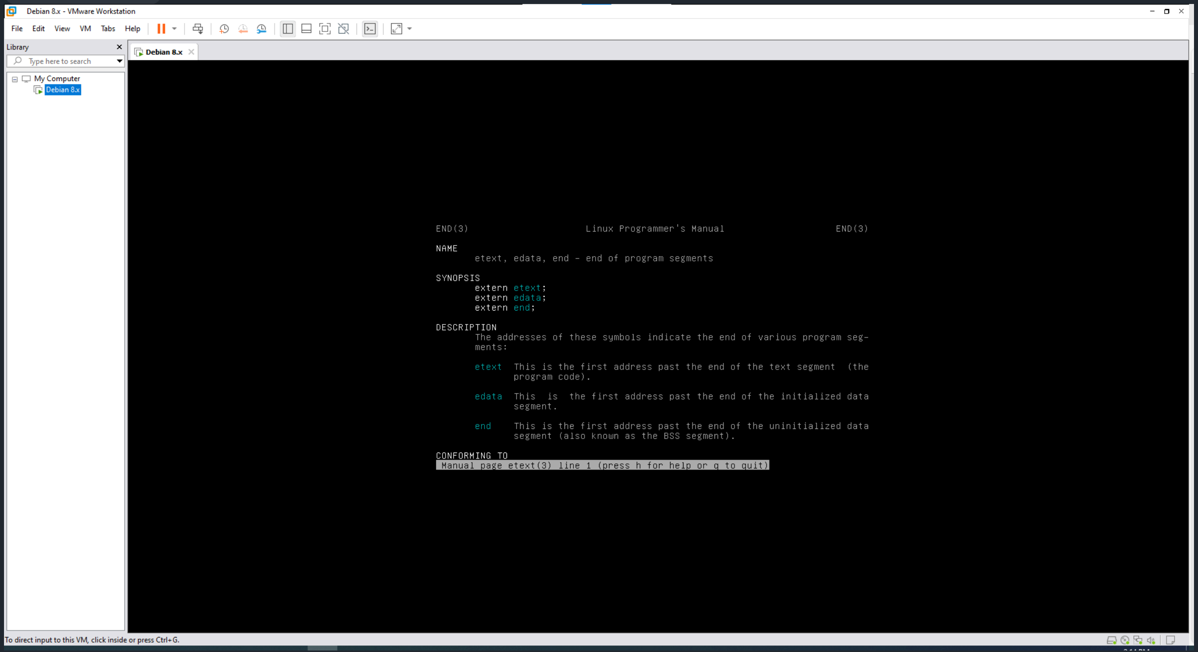


```
root@debian:~# ldd /bin/mkdir
        linux-gate.so.1 (0xb77d5000)
        libselinux.so.1 => /lib/i386-linux-gnu/libselinux.so.1 (0xb77a0000)
        libc.so.6 => /lib/i386-linux-gnu/i686/cmov/libc.so.6 (0xb75f3000)
        libpcre.so.3 => /lib/i386-linux-gnu/libpcre.so.3 (0xb757f000)
        libdl.so.2 => /lib/i386-linux-gnu/i686/cmov/libdl.so.2 (0xb757a000)
        /lib/ld-linux.so.2 (0xb77d6000)
        libpthread.so.0 => /lib/i386-linux-gnu/i686/cmov/libpthread.so.0 (0xb755
e000)
root@debian:~#
```

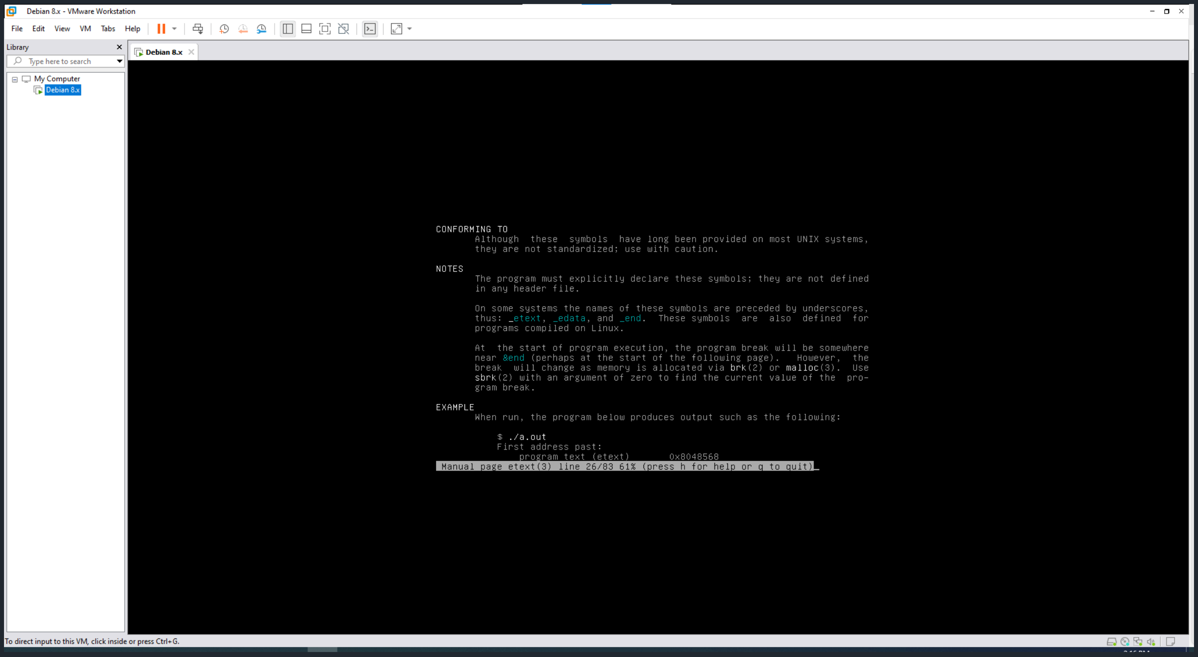
To direct input to this VM, click inside or press Ctrl+G.

☑ Getting started with addresses

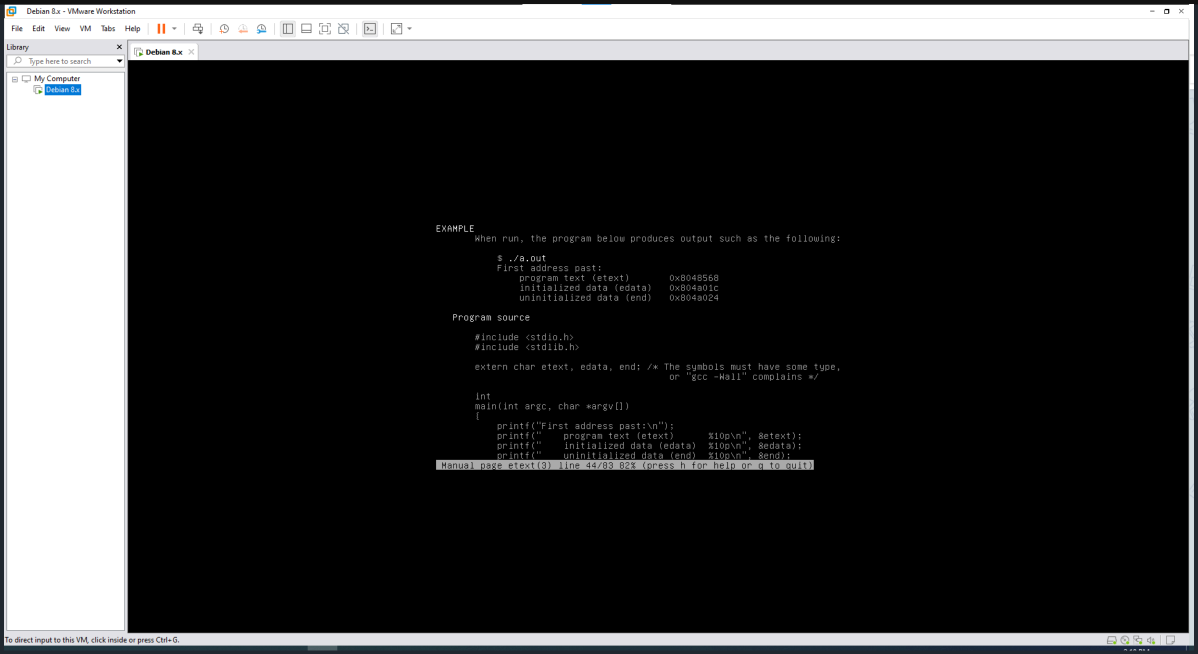
☑ `man etext`


Debian 8.x - VMware Workstation
File Edit View VM Tabs Help
Library
Type here to search
My Computer
Debian 8.x
To direct input to this VM, click inside or press Ctrl+G.

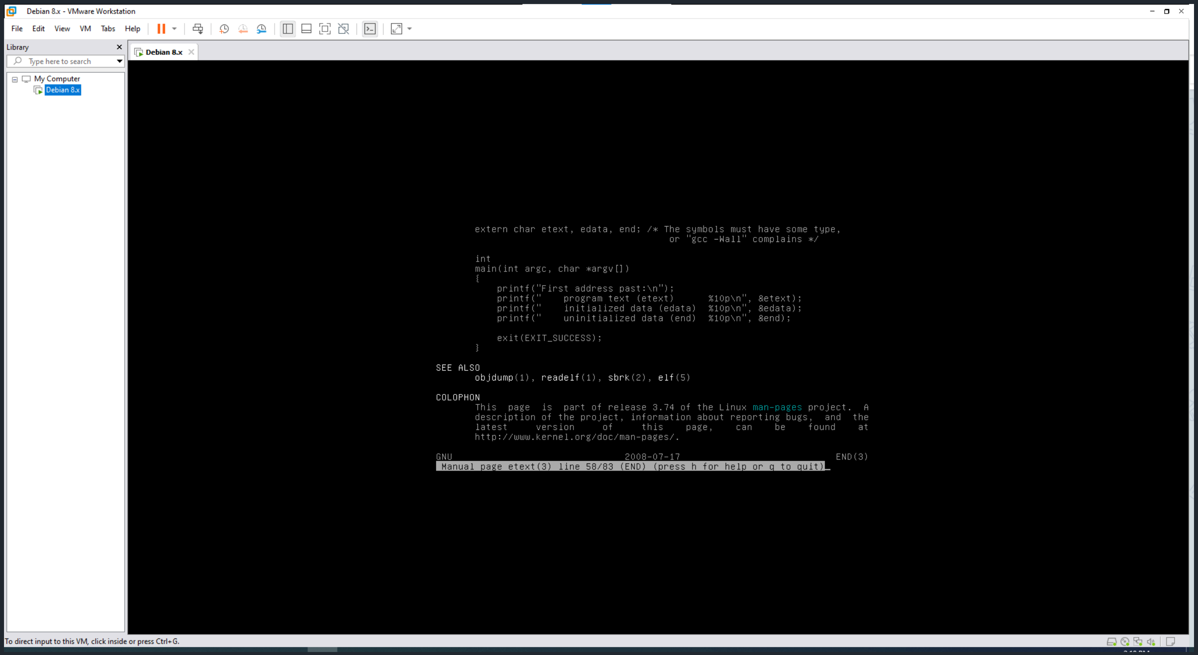
```
END(3)                     Linux Programmer's Manual                    END(3)

NAME
       etext, edata, end - end of program segments

SYNOPSIS
       extern etext;
       extern edata;
       extern end;

DESCRIPTION
       The addresses of these symbols indicate the end of various program seg‐
       ments:

       etext  This is the first address past the end of the text segment  (the
              program code).

       edata  This  is  the first address past the end of the initialized data
              segment.

       end    This is the first address past the end of the uninitialized data
              segment (also known as the BSS segment).

CONFORMING TO
 Manual page etext(3) line 1 (press h for help or q to quit)
```


Debian 8.x - VMware Workstation
File Edit View VM Tabs Help
Library
Type here to search
My Computer
Debian 8.x
To direct input to this VM, click inside or press Ctrl+G.

```
CONFORMING TO
       Although  these  symbols  have long been provided on most UNIX systems,
       they are not standardized; use with caution.

NOTES
       The program must explicitly declare these symbols; they are not defined
       in any header file.

       On some systems the names of these symbols are preceded by underscores,
       thus: _etext, _edata, and _end.  These symbols  are  also  defined  for
       programs compiled on Linux.

       At  the start of program execution, the program break will be somewhere
       near &end (perhaps at the start of the following page).   However,  the
       break  will change as memory is allocated via brk(2) or malloc(3).  Use
       sbrk(2) with an argument of zero to find the current value of the  pro‐
       gram break.

EXAMPLE
       When run, the program below produces output such as the following:

           $ ./a.out
           First address past:
               program text (etext)       0x8048568
 Manual page etext(3) line 26/83 61% (press h for help or q to quit)
```

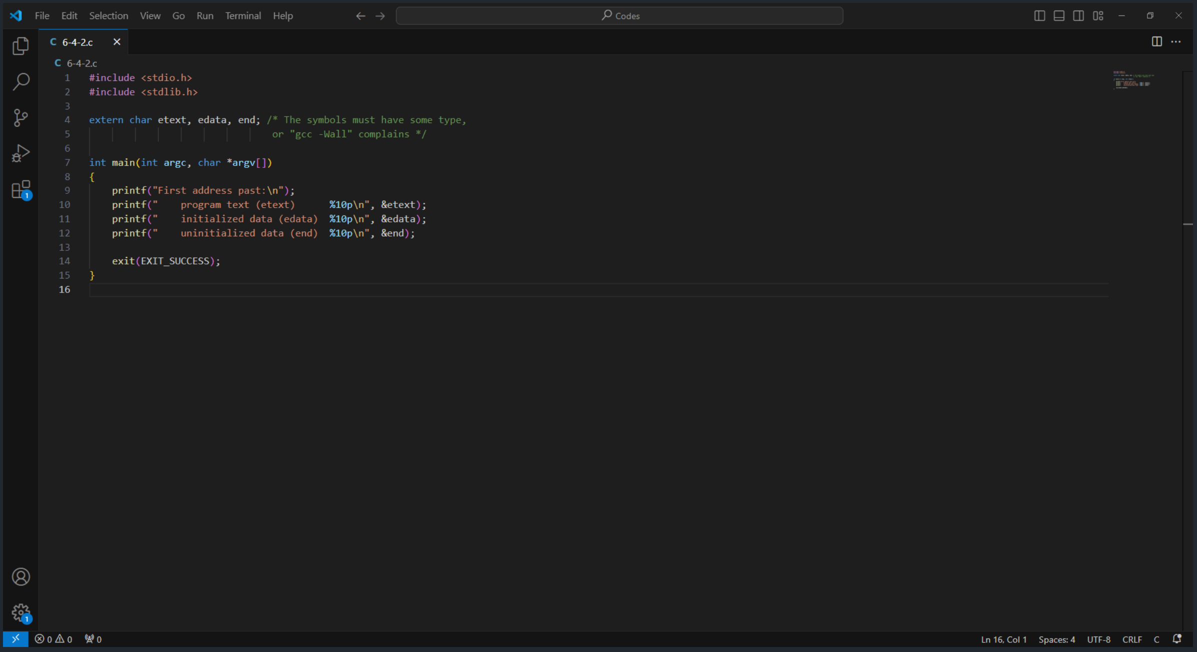

Debian 8.x - VMware Workstation
File Edit View VM Tabs Help
Library
Type here to search
My Computer
Debian 8.x
Debian 8.x
To direct input to this VM, click inside or press Ctrl+G.

```
EXAMPLE
       When run, the program below produces output such as the following:

           $ ./a.out
           First address past:
               program text (etext)       0x8048568
               initialized data (edata)   0x804a01c
               uninitialized data (end)   0x804a024

   Program source

       #include <stdio.h>
       #include <stdlib.h>

       extern char etext, edata, end; /* The symbols must have some type,
                                          or "gcc -Wall" complains */

       int
       main(int argc, char *argv[])
       {
           printf("First address past:\n");
           printf("    program text (etext)      %10p\n", &etext);
           printf("    initialized data (edata)  %10p\n", &edata);
           printf("    uninitialized data (end)  %10p\n", &end);
 Manual page etext(3) line 44/83 82% (press h for help or q to quit)
```


Debian 8.x - VMware Workstation
File Edit View VM Tabs Help
Library
Type here to search
My Computer
Debian 8.x
Debian 8.x
To direct input to this VM, click inside or press Ctrl+G.

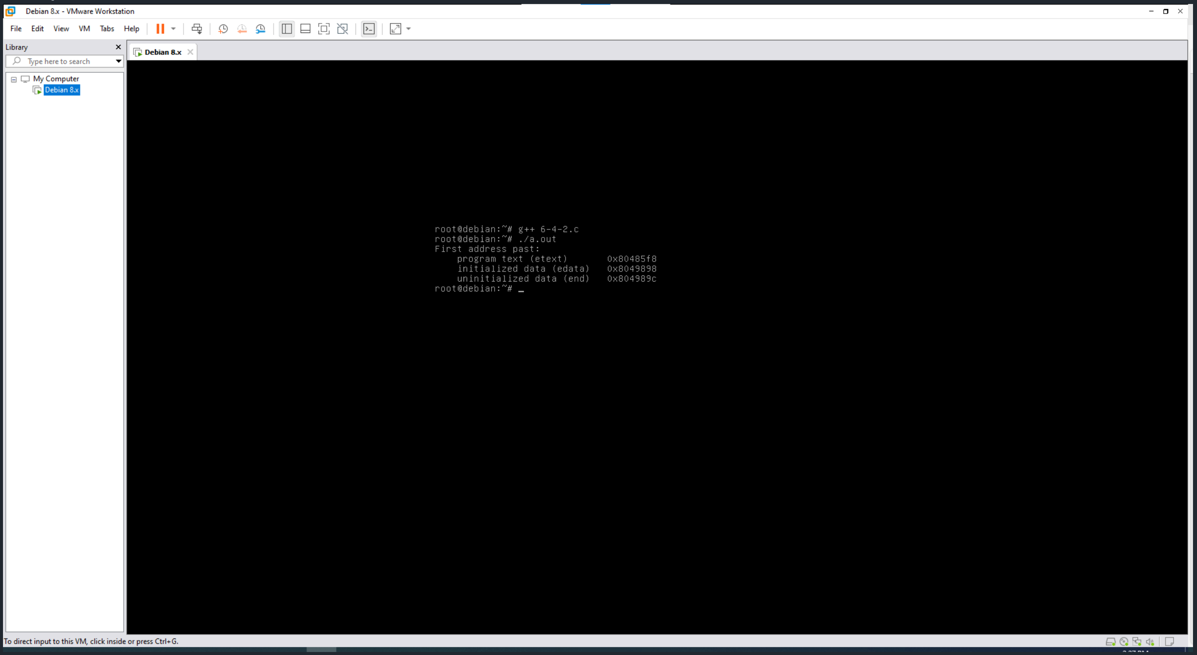
```
       extern char etext, edata, end; /* The symbols must have some type,
                                          or "gcc -Wall" complains */

       int
       main(int argc, char *argv[])
       {
           printf("First address past:\n");
           printf("    program text (etext)      %10p\n", &etext);
           printf("    initialized data (edata)  %10p\n", &edata);
           printf("    uninitialized data (end)  %10p\n", &end);

           exit(EXIT_SUCCESS);
       }

SEE ALSO
       objdump(1), readelf(1), sbrk(2), elf(5)

COLOPHON
       This  page  is  part of release 3.74 of the Linux man-pages project.  A
       description of the project, information about reporting bugs,  and  the
       latest    version    of    this    page,    can    be    found    at
       http://www.kernel.org/doc/man-pages/.

GNU                              2008-07-17                         END(3)
 Manual page etext(3) line 58/83 (END) (press h for help or q to quit)
```

- ☑ Code (in file 6-4-2.c):


File Edit Selection View Go Run Terminal Help
Codes
6-4-2.c
Ln 16, Col 1 Spaces: 4 UTF-8 CRLF C

```c
#include <stdio.h>
#include <stdlib.h>

extern char etext, edata, end; /* The symbols must have some type,
                                  or "gcc -Wall" complains */

int main(int argc, char *argv[])
{
    printf("First address past:\n");
    printf("    program text (etext)      %10p\n", &etext);
    printf("    initialized data (edata)  %10p\n", &edata);
    printf("    uninitialized data (end)  %10p\n", &end);

    exit(EXIT_SUCCESS);
}
```

Output:


Debian 8.x - VMware Workstation
File Edit View VM Tabs Help
Library
Type here to search
My Computer
Debian 8.x
To direct input to this VM, click inside or press Ctrl+G.

```
root@debian:~# g++ 6-4-2.c
root@debian:~# ./a.out
First address past:
    program text (etext)        0x80485f8
    initialized data (edata)    0x8049898
    uninitialized data (end)    0x804989c
root@debian:~# _
```

As you see, the addresses are exactly the same as what we expected.

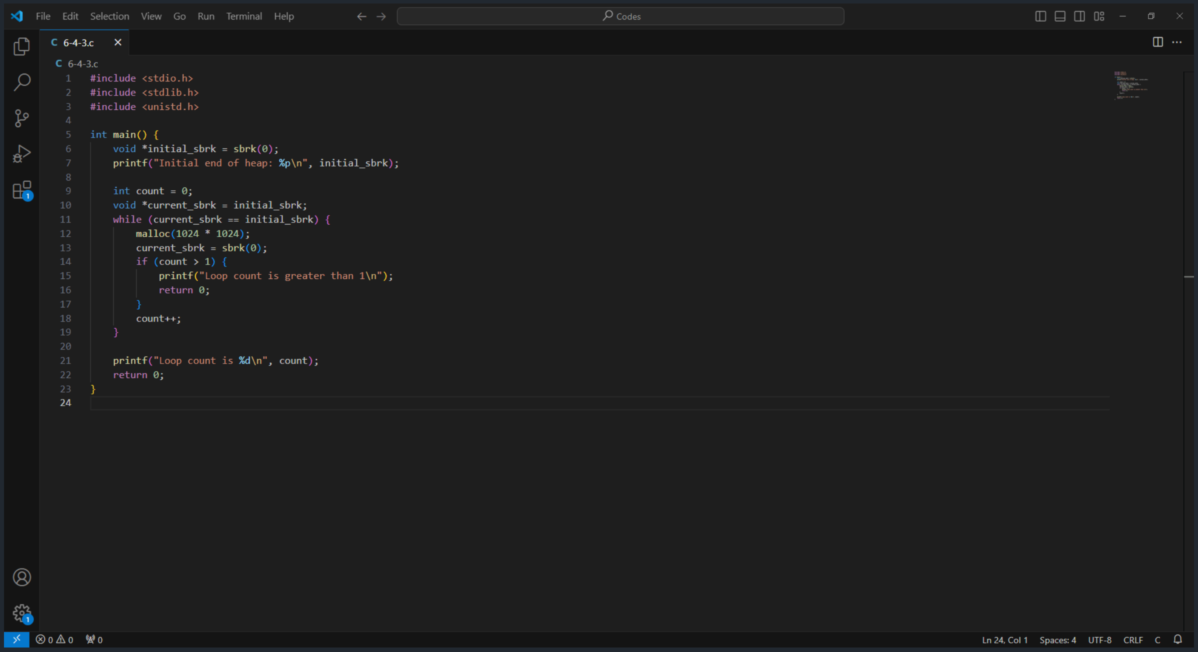
- ☑ Description about `etext`, the meaning of the comment, why using the term symbol, why using extern:
  - ☑ The symbols are:
    - ☑ `etext`: This is the first address past the end of the text segment (the program code). The text segment contains the compiled machine code of the program.
    - ☑ `edata`: This is the first address past the end of the initialized data segment
    - ☑ `end`: This is the first address past the end of the uninitialized data segment (also known as the BSS segment).
  - ☑ Meaning of the comment: The '-Wall' flag enables all compiler's warning messages, which helps in writing cleaner and more error-free code. In C, declaring symbols with extern

requires specifying a type. Without a type, the compiler would issue warnings or errors. Here, char is chosen as the type because it is a basic type that simply gives these symbols a valid type without any specific size or alignment constraints.

- ☑ Why use the term symbol: These symbols () are used to refer to specific locations in memory and do not hold values themselves. They are part of the program's memory map, which is why they are called symbols rather than variables.
- ☑ Why use extern: The extern keyword is used to declare these symbols without defining them. This informs the compiler that these symbols are defined elsewhere, typically by the linker. These symbols are defined by the linker, not by the C code, so using extern tells the compiler that their actual addresses will be resolved during the linking stage. Also, by declaring them as extern char, the code can take the address of these symbols (using &etext, &edata, &end) to print or otherwise.

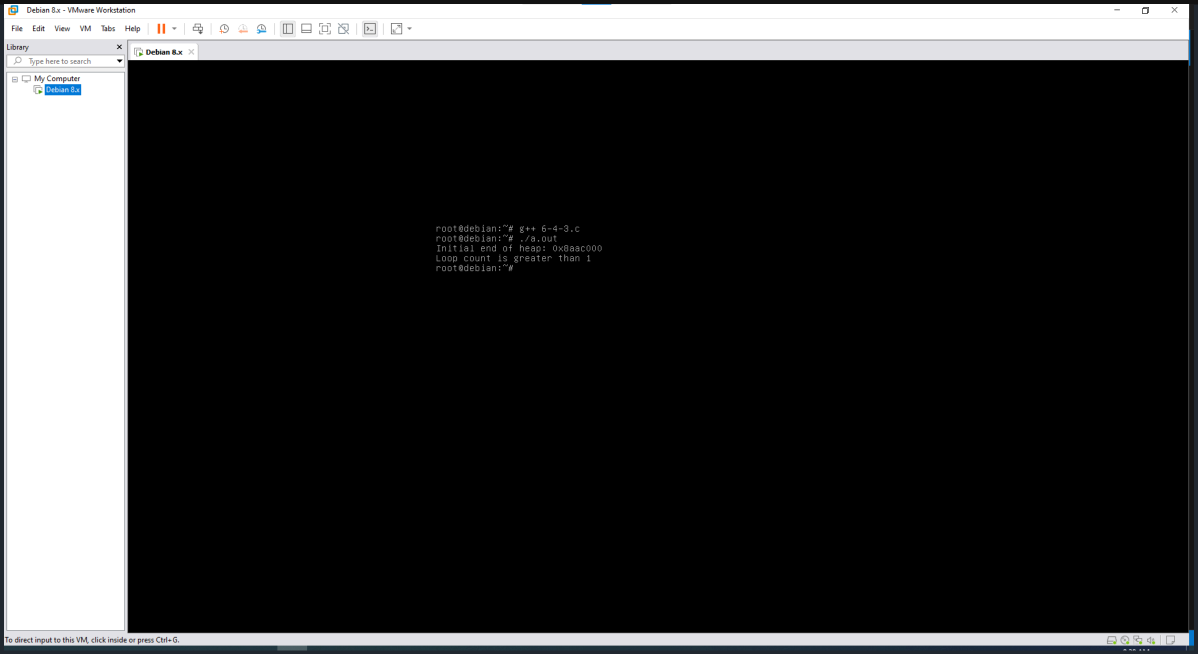
☑ sbrk analysis:

Code (in file 6-4-3.c):



```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

int main() {
    void *initial_sbrk = sbrk(0);
    printf("Initial end of heap: %p\n", initial_sbrk);

    int count = 0;
    void *current_sbrk = initial_sbrk;
    while (current_sbrk == initial_sbrk) {
        malloc(1024 * 1024);
        current_sbrk = sbrk(0);
        if (count > 1) {
            printf("Loop count is greater than 1\n");
            return 0;
        }
        count++;
    }

    printf("Loop count is %d\n", count);
    return 0;
}
```

Output:


Debian 8.x - VMware Workstation
File Edit View VM Tabs Help
Library
Type here to search
My Computer
Debian 8.x
Debian 8.x
To direct input to this VM, click inside or press Ctrl+G.

```
root@debian:~# g++ 6-4-3.c
root@debian:~# ./a.out
Initial end of heap: 0x8aac000
Loop count is greater than 1
root@debian:~#
```

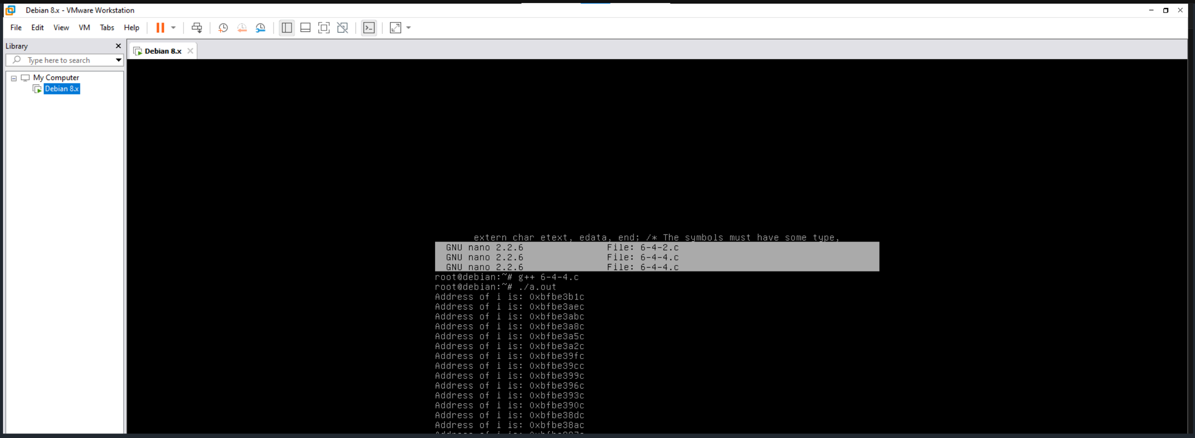
As you see, the loop count is not equal to 1. That's because memory allocators often request larger chunks of memory from the operating system and then manage these chunks internally. When malloc is called, it might be allocating memory from a pre-existing pool rather than extending the heap. This means the end of the heap doesn't change with every call to malloc. In the above code, the size of malloc was 1 MB. If we change it to 1 KB, the address of end of heap will change and the loop count will be 1. Its output will be:

We could write code like this to see if malloc changes the address of end of heap:

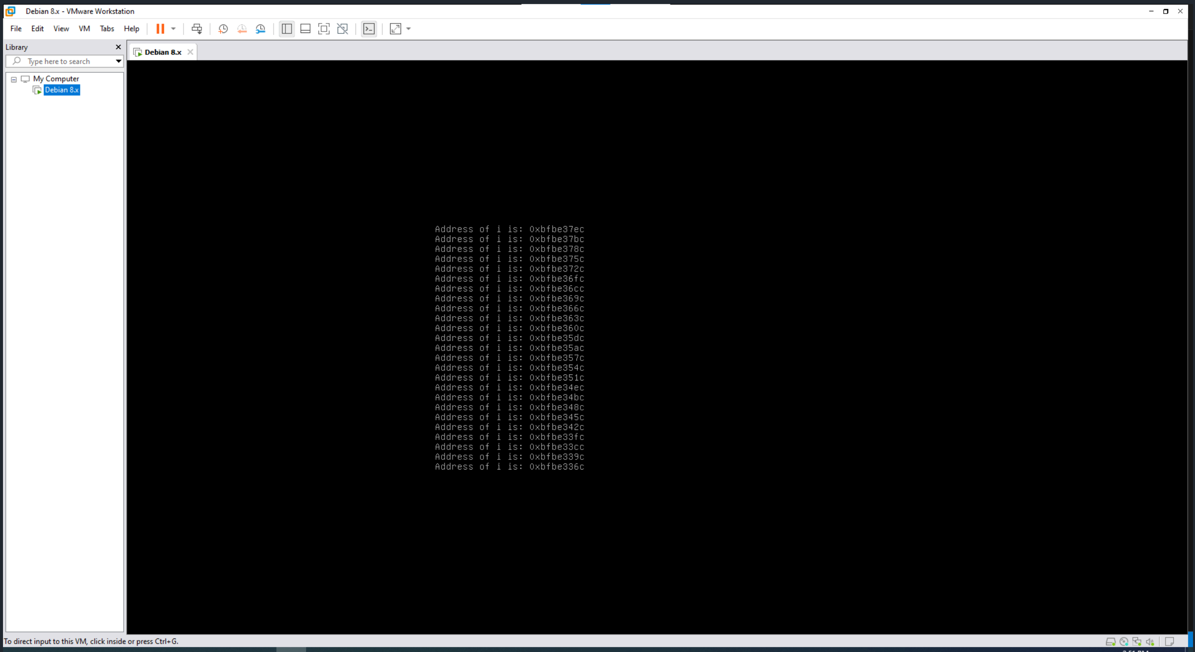
```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

int main() {
    printf("End of heap before malloc: %p\n", sbrk(0));
    malloc(1024 * 1024);
    printf("End of heap after malloc: %p\n", sbrk(0));
    return 0;
}
```

☑ stack growth analysis:
Code (in file 6-4-4.c):

Output:

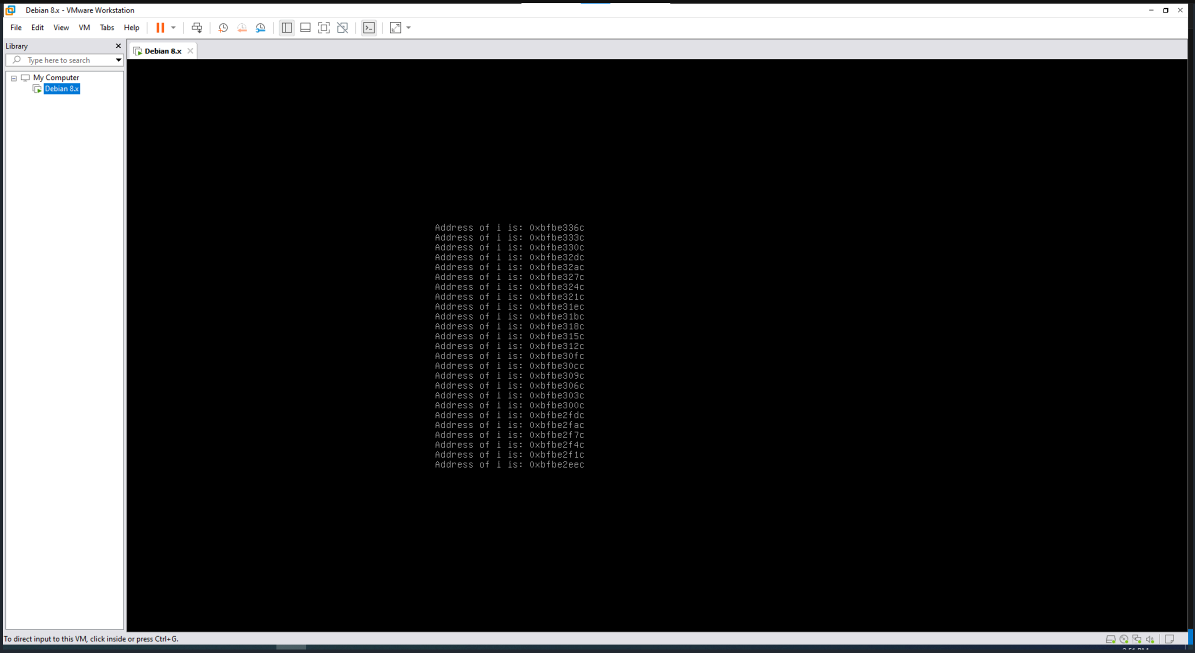


```
     extern char etext, edata, end; /* The symbols must have some type,
  GNU nano 2.2.6              File: 6-4-2.c
  GNU nano 2.2.6              File: 6-4-4.c
  GNU nano 2.2.6              File: 6-4-4.c
root@debian:~# g++ 6-4-4.c
root@debian:~# ./a.out
Address of i is: 0xbfbe3b1c
Address of i is: 0xbfbe3aec
Address of i is: 0xbfbe3abc
Address of i is: 0xbfbe3a8c
Address of i is: 0xbfbe3a5c
Address of i is: 0xbfbe3a2c
Address of i is: 0xbfbe39fc
Address of i is: 0xbfbe39cc
Address of i is: 0xbfbe399c
Address of i is: 0xbfbe396c
Address of i is: 0xbfbe393c
Address of i is: 0xbfbe390c
Address of i is: 0xbfbe38dc
Address of i is: 0xbfbe38ac
```



```
Address of i is: 0xbfbe37ec
Address of i is: 0xbfbe37bc
Address of i is: 0xbfbe378c
Address of i is: 0xbfbe375c
Address of i is: 0xbfbe372c
Address of i is: 0xbfbe36fc
Address of i is: 0xbfbe36cc
Address of i is: 0xbfbe369c
Address of i is: 0xbfbe366c
Address of i is: 0xbfbe363c
Address of i is: 0xbfbe360c
Address of i is: 0xbfbe35dc
Address of i is: 0xbfbe35ac
Address of i is: 0xbfbe357c
Address of i is: 0xbfbe354c
Address of i is: 0xbfbe351c
Address of i is: 0xbfbe34ec
Address of i is: 0xbfbe34bc
Address of i is: 0xbfbe348c
Address of i is: 0xbfbe345c
Address of i is: 0xbfbe342c
Address of i is: 0xbfbe33fc
Address of i is: 0xbfbe33cc
Address of i is: 0xbfbe339c
Address of i is: 0xbfbe336c
```

To direct input to this VM, click inside or press Ctrl+G.


Debian 8.x - VMware Workstation
File Edit View VM Tabs Help
Library
Type here to search
My Computer
Debian 8.x
Debian 8.x
To direct input to this VM, click inside or press Ctrl+G.

```
Address of i is: 0xbfbe336c
Address of i is: 0xbfbe333c
Address of i is: 0xbfbe330c
Address of i is: 0xbfbe32dc
Address of i is: 0xbfbe32ac
Address of i is: 0xbfbe327c
Address of i is: 0xbfbe324c
Address of i is: 0xbfbe321c
Address of i is: 0xbfbe31ec
Address of i is: 0xbfbe31bc
Address of i is: 0xbfbe318c
Address of i is: 0xbfbe315c
Address of i is: 0xbfbe312c
Address of i is: 0xbfbe30fc
Address of i is: 0xbfbe30cc
Address of i is: 0xbfbe309c
Address of i is: 0xbfbe306c
Address of i is: 0xbfbe303c
Address of i is: 0xbfbe300c
Address of i is: 0xbfbe2fdc
Address of i is: 0xbfbe2fac
Address of i is: 0xbfbe2f7c
Address of i is: 0xbfbe2f4c
Address of i is: 0xbfbe2f1c
Address of i is: 0xbfbe2eec
```

As you see, addresses decrease because the stack grows backward (from higher address to lower address).

The codes are:
Codes.zip

BozorgmehrZia added the documentation label on Aug 4

BozorgmehrZia assigned AMshoka on Aug 4

Assignees

AMshoka

Labels

documentation

Projects

None yet

Milestone

No milestone

Development

Create a branch for this issue or link a pull request.

2 participants

Pin issue