Example Output

```
init: starting sh
$ myvmstats
Connor Palmira, xv6 Memory Stats
Total memory: 134217728
Free memory: 133251072
Used memory: 966656
Resident pages memory: 12288
Process memory: 16384

Memory successfully allocated using malloc

Connor Palmira, xv6 Memory Stats
Total memory: 134217728
Free memory: 133115904
Used memory: 1101824
Resident pages memory: 147456
Process memory: 147472
$
```

Explanation of sys_vmstats()

The system call sys_vmstats() takes in 1 parameter for an address for the new struct that is defined in “vmstat.h”, vmstat. If that is not given as a parameter for the system call, sys_vmstats() returns -1. After checking the parameters, the system call then fills out the needed information for the vmstat struct. This information includes the system’s amount of total memory, free memory, and used memory in bytes, the amount of resident pages, and the amount of memory used for the current process in bytes. To get the free memory, the function get_free_mem_count() is called. To get the amount of resident pages, the function count_resident_page() is called with the pid of the current process as the parameter. After doing so, the copyout() function is called and if it returns with less than 0, then sys_vmstats() returns with -1. Otherwise, the system call ends and returns with 0.

Explanation of get_free_mem_count()

The function get_free_mem_count() counts the number of free memory pages and returns the result. It uses a pointer to struct run, which is a linked list of memory pages. It uses this pointer to go down the list of memory pages and increment the sum until the linked list ends. It does all of this in a spinlock. The reason that get_free_mem_count() is defined in “kalloc.c” is because “kalloc.c” is responsible for the allocation and deallocation of memory pages. By defining get_free_mem_count() in “kalloc.c”, the function is given the ability to access the same data concerning the memory that the other functions in “kalloc.c” are able to access.

Explanation of count_resident_pages()
The function count_resident_pages() counts the amount of memory pages for the given process. The given process is passed in via parameter. The function uses a for-loop to iterate over the address space of the given process. In the for-loop, the function walk() is called. If the current page entry being checked in the for-loop is found, is valid, and is in user mode, then the count of pages is incremented by one. After the address space of the given process is incremented over, the count of the resident pages is returned.

Explanation of copyout()
The function copyout(), defined in “vm.c” is a function that copies information from the kernel to the user. The function has four parameters. The first parameter is the page table for the process, which copyout() will be using to copy information. The second parameter is the address for copyout() to copy information to. The third parameter is a pointer to the data that needs to be copied. The fourth and final parameter is the number of bytes that needs to be copied. The function makes sure that the address is valid then translates the virtual address to a physical address. During this time, copyout() checks the permissions of the address space it is checking. If an error occurs at any point, the function returns -1, otherwise the function is successful and returns 0.

Difference before and after using malloc()
After using malloc(), the amount of free memory decreased while the amount of used memory, resident pages memory, and process memory all increased. The used memory and resident pages memory all increased by 135,168 bytes each. The amount of free memory decreased by 135,168 bytes. The amount of process memory increased by 131,088 bytes. The total amount of memory stayed the same (134,217,728 bytes) after using malloc(). Furthermore, the amount of free memory and used memory still adds up to the total amount of memory.