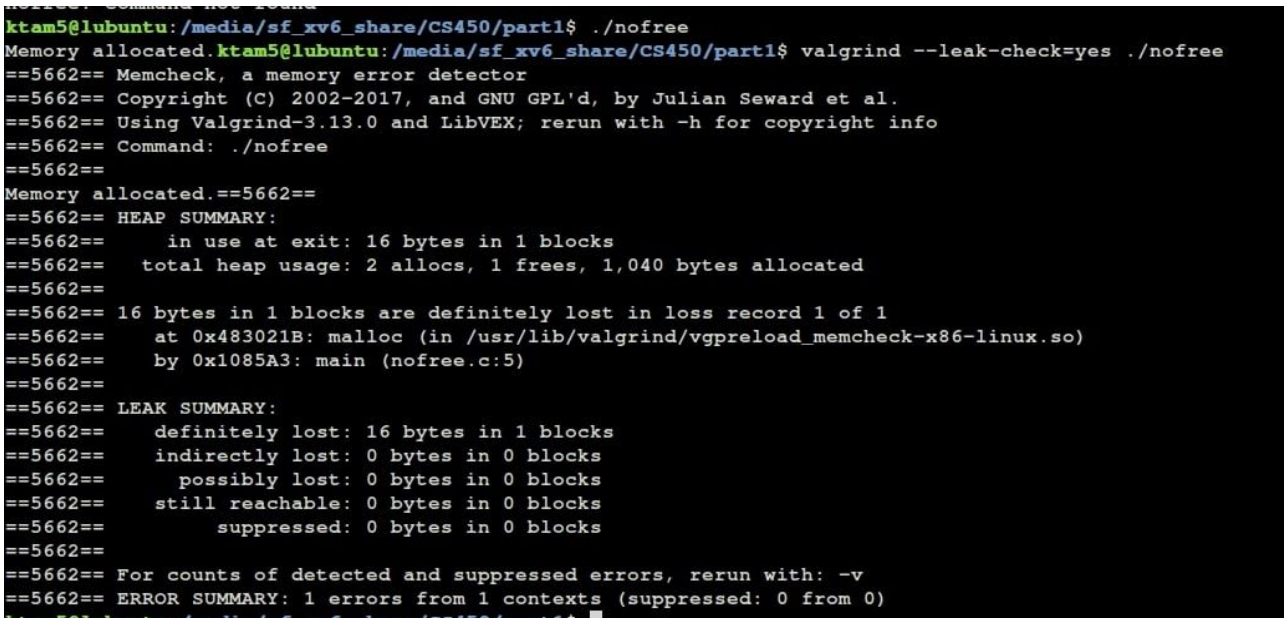
Ka Tam (A20374415)
Jack Critzer (A20396230)
CS 450
4/5/2020
Programming Assignment 3

Part 1 Documentation

Below is the result from part 1 of this assignment. To address the deliverable #1, we created the c program of malloc_no_free.c to demonstrate what it is like to not include free() for malloc().

Deliverable 1:
The program was able to run normally without any problems. Running the gdb alongside does not have any errors as well. Everything seems normal except when using valgrind. Below is the screenshot when using "valgrind --leak-check=yes" to run the program. The tool valgrind is able to show the amount of space that cannot be reached from the program. And in this case, there are 16 bytes that are definitely lost by not freeing the memory.

Malloc, No Free



Deliverable 2:

As for the second part of the deliverables, we were asked to find other test cases for valgrind. With memory allocation in mind, we decided to explore the difference in valgrind execution when dealing with malloc and free. The first one demonstrates the normal malloc and free. Resulting in no error being found in both gdb and valgrind.

Malloc, Free

```
jack@jack-VirtualBox:~/Documents/xv6-shared/CS450/part1$ valgrind --leak-check=yes ./malloc_free
==10800== Memcheck, a memory error detector
==10800== Copyright (C) 2002-2015, and GNU GPL'd, by Julian Seward et al.
==10800== Using Valgrind-3.11.0 and LibVEX; rerun with -h for copyright info
==10800== Command: ./malloc_free
==10800==
Memory allocated.==10800==
==10800== HEAP SUMMARY:
==10800==     in use at exit: 0 bytes in 0 blocks
==10800==   total heap usage: 2 allocs, 2 frees, 1,040 bytes allocated
==10800==
==10800== All heap blocks were freed -- no leaks are possible
==10800==
==10800== For counts of detected and suppressed errors, rerun with: -v
==10800== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

The next one we have to consider is no malloc but call free(). Valgrind shows that there is some error in the program. The error message is “Invalid free() / delete / delete[] / realloc()”. It points to the line number in which the free() was called. With this error message, we can easily find where the mistakes are.

No Malloc, Free

```
jack@jack-VirtualBox:~/Documents/xv6-shared/CS450/part1$ valgrind --leak-check=yes ./no_malloc_free
==11000== Memcheck, a memory error detector
==11000== Copyright (C) 2002-2015, and GNU GPL'd, by Julian Seward et al.
==11000== Using Valgrind-3.11.0 and LibVEX; rerun with -h for copyright info
==11000== Command: ./no_malloc_free
==11000==
==11000== Invalid free() / delete / delete[] / realloc()
==11000==    at 0x482D358: free (in /usr/lib/valgrind/vgpreload_memcheck-x86-linux.so)
==11000==    by 0x108643: main (no_malloc_free.c:17)
==11000==  Address 0xbef18f54 is on thread 1's stack
==11000==  in frame #1, created by main (no_malloc_free.c:4)
==11000==
0xbef18f54==11000==
==11000== HEAP SUMMARY:
==11000==     in use at exit: 0 bytes in 0 blocks
==11000==   total heap usage: 1 allocs, 2 frees, 1,024 bytes allocated
==11000==
==11000== All heap blocks were freed -- no leaks are possible
==11000==
==11000== For counts of detected and suppressed errors, rerun with: -v
==11000== ERROR SUMMARY: 1 errors from 1 contexts (suppressed: 0 from 0)
```

The last case for malloc free is no malloc and no free. All the variables are stored in the stack of the address space. There is nothing that needs to be dynamically allocated. Thus, there won’t be any error for not using malloc and not freeing.

No Malloc, No Free

```
jack@jack-VirtualBox:~/Documents/xv6-shared/CS450/part1$ valgrind --leak-check=yes ./no_malloc_no_free
==10956== Memcheck, a memory error detector
==10956== Copyright (C) 2002-2015, and GNU GPL'd, by Julian Seward et al.
==10956== Using Valgrind-3.11.0 and LibVEX; rerun with -h for copyright info
==10956== Command: ./no_malloc_no_free
==10956==
0xbed26f54==10956==
==10956== HEAP SUMMARY:
==10956==     in use at exit: 0 bytes in 0 blocks
==10956==   total heap usage: 1 allocs, 1 frees, 1,024 bytes allocated
==10956==
==10956== All heap blocks were freed -- no leaks are possible
==10956==
==10956== For counts of detected and suppressed errors, rerun with: -v
==10956== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

Another test case is trying to access the allocated area after freeing it. The program will run normally without any problems. But we know that it is not allowed to access the memory after freeing it. After running valgrind, there display an error saying “invalid write of size 4” and the line number of the code that causes this error. This can be great for debugging because we know exactly where it is happening.

Access after free

```
ktam5@lubuntu:/media/sf_xv6_share/CS450/part1$ gcc -Wall -Werror -g access_after_free.c -o access_after_free
ktam5@lubuntu:/media/sf_xv6_share/CS450/part1$ ./access_after_free
Memory allocated.ktam5@lubuntu:/media/sf_xv6_shvalgrind --leak-check=yes ./access_after_free
==7419== Memcheck, a memory error detector
==7419== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==7419== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
==7419== Command: ./access_after_free
==7419==
==7419== Invalid write of size 4
==7419==    at 0x10864C: main (access_after_free.c:23)
==7419==  Address 0x4a2f02c is 4 bytes inside a block of size 16 free'd
==7419==    at 0x48313D7: free (in /usr/lib/valgrind/vgpreload_memcheck-x86-linux.so)
==7419==    by 0x108642: main (access_after_free.c:21)
==7419==  Block was alloc'd at
==7419==    at 0x483021B: malloc (in /usr/lib/valgrind/vgpreload_memcheck-x86-linux.so)
==7419==    by 0x1085D3: main (access_after_free.c:6)
==7419==
Memory allocated.==7419==
==7419== HEAP SUMMARY:
==7419==     in use at exit: 0 bytes in 0 blocks
==7419==   total heap usage: 2 allocs, 2 frees, 1,040 bytes allocated
==7419==
==7419== All heap blocks were freed -- no leaks are possible
==7419==
==7419== For counts of detected and suppressed errors, rerun with: -v
==7419== ERROR SUMMARY: 1 errors from 1 contexts (suppressed: 0 from 0)
ktam5@lubuntu:/media/sf_xv6_share/CS450/part1$
```