CS444 Homework 3

Lilliana Watts, Nicholas Skinner, Yong Ping Li

June 7, 2018

DESIGN PLAN

In starting our design plans, we decided to look at the base SLOB.c as well as do some reading and research to gain a better understanding of what it is doing, and what are the core components of it. We had decided that it would be a good idea to implement a system call to display our test results through a system call, so we would need to do some research to implement that as well.

WRITEUP QUESTIONS

A. *What do you think the main point of this assignment is?*

• The main point of this assignment is to have our teams research and understand how memory is managed, and how it is allocated in the SLOB allocator. We are also intended to learn how to implement our own system calls, and then use them to test our implementation in the kernel.

B. *How did you personally approach the problem? Design decisions, algorithm, etc.*

• We started by examining the existing SLOB.c implementation, and we would look to modify its page allocation to search for the best fit in the free memory, rather than the first possible fit. We did so by iterating through pages in memory, and saving the relevant data to any location that had less wasted memory than the last best fit. We would iterate until either an exact fit had been found, or when we reached the last possible candidate location. If no location could be found with enough memory requested, we would have the function return NULL.

• After Finding the best fit, we had reused the existing code for allocation after reassigning the best values to those that would have previously been the first fit values.

C. How did you ensure your solution was correct? Testing details, for instance.

• To ensure the solution was correct, we had designed a system call in order to report the amount of memory in use, and then made a c file that calls this system call.

• Clone the kernel code to a fresh instance of a repo: git clone git://git.yoctoproject.org/linux-yocto-3.19

• Navigate into the Git repo folder: cd linux-yocto-3.19 • Checkout the appropriate version of our kernel: git checkout v3.19.2

• Apply the patch file : git apply mm.patch

• Apply the patch file : git apply include.patch

• Apply the patch file : git apply Sys.patch

• Source the proper environment for the kernel: source

/scratch/files/environment-setup-i586-poky-linux.csh

• Copy the core into the linux folder: cp

/scratch/files/core-image-lsb-sdk-qemux86.ext4 .

• Copy the config file into the linux folder: cp

/scratch/files/config-3.19.2-yocto-standard .config

• Go into the menuconfig, and then set the SLOB to be active into a module:

make menuconfig

Go to General Setup

Under Choose SLAB allocator, select SLOB block drivers

Save the new configuration, and then exit

• After setting the menuconfig, now we will make the VM: make -j4 all

• next we will start the VM with networking enabled: qemu-system-i386 -gdb tcp::5507 -S -nographic -kernel bzImage-qemux86.bin -drive file=core-image-lsb-sdk-qemux86.ext4,if=virtio -enable-kvm -usb -localtime –no-reboot –append "root=/dev/vda rw console=ttyS0 debug".

• Login with the root account, there is no password, so just type: root

• we want to copy our fragment file here using whatever method we can, I used scp (see below) • We will now Compile the fragment file using

• gcc Fragment -o out

• Run the fragment file ./out

• Copy our testing file using SCP:

```
scp skinnern@os2.engr.oregonstate.edu:
/scratch/spring2018/group7_new/SLOBPATCH/VerificationScripts/
Reg/Fragment.c .
```

D. What did you learn?

• We learned how system calls are implemented within the linux kernel, as well as how those calls are invoked within the user space. We also learned how a slab allocator searches for available memory, and how it allocates the memory to its kernel objects.

GITHUB LOG

Detail	Author	Description
77f914a	Nicholas	Placeholder

WORK LOG

Date	Name	Hours	Description
6/6	Nicholas	1	Setup Document
6/7	Nicholas	3	Research SLOB
6/7	Nicholas	3	Write Best SLOB
6/7	Nicholas	2	Make patches and test
6/7	Nicholas	1	organize and tarball

REFERENCES

[1] Jonathan Corbet, Alessandro Rubini, and Greg Kroah-Hartman. *Linux Device Drivers*. O'Reilly Media, Inc., 3 edition, 2005.