# Assignment 4

Shawn Cross, Ryan Crane

December 1, 2017

## 1 Plan for Implementation

For this project we plan our plan is to try and first find a way to apply the best fit solution and then we will use the system calls to determine the fragmentation that we are getting with this solution. We will then compare that to the fragmentation of the first fit solution.

## 2 Version Control Log

| Detail | Author | Description |
|---|---|---|
| 77ef891 | Shawn Cross | adding concurrency 3 files. |
| 493293c | Shawn Cross | adding last files for concurrency 3 |
| a36cf1d | Shawn Cross | adding the title page to the tex document |
| a488836 | Shawn Cross | fixed small problem |
| 99fd75b | Shawn Cross | Adding assignment 4 forgot to commit the previous changes. |

## 3 Work Log

Tuesday November 28th 10 - 11:30am Began the assignment modified slob.c with a best fit solution, but couldn't make the kernel run it. Wednesday November 29th 1:30pm - 3:30pm Went to office hours for questions, got kernel to load slob code. Thursday November 30th 8:30am - 11:30am implemented best fit in the kernel and wrote the system calls. Began the test program to check memory use. Friday December 1st 9:00am - 4:00pm Finished testing program and wrote tex document.

## 4 Project Questions

### 4.1 Main point of the assignment

The main point of this assignment is to teach us about how memory is managed in the linux kernel as well as teach us that some times a solution that sounds the best is not always the right solution.

### 4.2 How did we approach the problem

We knew we needed to edit the current slob file to use best fit instead of first fit. So we decide to try and figure out how to do this first. We came up with a simple solution. Set our page struct equal to the orignal one and then check if the next one is smaller if it is set our equal to it. Then check if the current spot is equal to out chunk and if it is put it in that spot if not then try the next spot. Continue doing this until we find a spot that is either the same or the next page is NULL. In this case you would put it in the next best spot or create a new page. Then for the system call we just find the number of pages being used and then multiple it by the size of a page. For the free space we loop thorough all the pages finding all the free units and then add them together. We then us these to find the percentage of fragmentation.

### 4.3 Ensure correctness and details of how to preform

## 5 TA Evaluation

Steps for applying patch to clean linux-yocto-3.19 tree.

1. step 1: clone new repo from github using the command

   git clone git://git.yoctoproject.org/linux-yocto-3.19

2. step 2: switch to the 3.19.2 by using the command

   git checkout -b linux-yocto-3.19.2

3. step 3: copy the file /scratch/fall2017/files/config-3.19.2-yocto-qemu into a file named .config into the source root of the repo that you just cloned.

4. step 4: Now source the environment configuration. If you are using bash you will need to source /scratch/fall2017/files/environment-setup-i586-poky-linux and if you are using tcsh you will need to source /scratch/fall2017/files/environment-setup-i586-poky-linux.csh

5. step 5: You need to apply the patch we provided to the repo by using the command

   git apply [path to patch file] where path to patch file is the path you need to the patch file that we have provided. We have provided two patches firstfit.patch will be for the original slob that comes with the fresh kernel and then adds the system call function for testing. The bestfit.patch is our best fit solution along with the system call stuff as well. You can apply either wbut you will have to go back and clone a new repo before testing the other.

6. step 6: Yow will need to now run the command make menuconfig. Once this is done a menu will open and you will need to scroll down and change from using slub to using slob. This is in general setup towards the botton in the option choose slab allocator.

7. step 7: You now need to make the kernel by running the command make -j4

8. step 8: Once the kernel is created you will need to run the vm using the command

   qemu-system-i386 -gdb tcp::5550 -S -nographic -kernel [path to bzImage] -drive file=[path to core image],if=virtio -enable-kvm -usb -localtime –no-reboot –append ”root=/dev/vda rw console=ttyS0 debug”

   Where [path to core image] is the file found in

   /scratch/fall2017/files/core-image-lsb-sdk-qemux86.ext4

   and the [path to bzImage] for me would be the bzImage found in

   linux-yocto-3.19/arch/x86/boot/bzImage

   and the ???? are the port number that you want to connect gdb to remotely.

   After this command is run the terminal will halt.

9. step 9: you will now need to run $GDB in a new terminal window. You may need soucre the enviroment again as you did in step 4 previously if you did start a new termial session and the $GDB command does not work. Once gdb is running you will use the command

   target remote :????

   where the ???? are the same as the port that you had specified in step 7.

then type continue and the first terminal should start.

10. step 10: You need to then scp the the page_test.c and the mt19937.h files that we provided over to your running vm

11. step 11: You will then need to comiple this program by using the command: gcc page_test.c then you can run it by running: ./a.out

    this will output a bunch of precentages. These precentages are the amount of free free space avalible. The program mallocs and free's a bunch of random size arrays. You will notice that as the malloc's and free's happen you will see that we the precentages are increasing and decreasing. This goes along with the idea that as we are using more space the amout of free space will decreas and as we free the amount of space will increase. There are some random flucuation we believe this is because as pages are added or removed the amount of free space would increase quite a bit from this.

12. step 12: Now you will need to repeat theses steps to use the other patch that we have provided. You can either revert the repo back to its original state and then apply the other patch or you can download a fresh repo. Once you have does this you can start from either step on if you want a fresh repo or step two if you reverted the repo.

13. Best fit Vs First fit: One thing we did notice is that with the first fit solution the percentage of free space is much less than what we got for when we applied our best fit soultion. we are not exactly sure why this is but we did notice that our solution is flucuating the same way that the first fit is. We believe that this shows that our solution is in fact working and that the space is being allocated the way that it is supposed to be.

## 5.1 What did we learn

During this project we learned that while some solution seem to be the better idea it doesn't mean that they are the better way to do it. while first fit does seem to use a little more space and leaves more small holes in each page it also doesn't take as long to do. Our version of best fit causes adding and freeing space to take a lot more time to a point where the outcome is not a benifit.