MODIFIED CODE SECTION

(1) **Porting the Executable (echo, cat, wc)**

- I did not make a lot of changes to this code. I ported it from the xv6 OS with some minor changes to the headings. I then compiled them into executables using this syntax: gcc –o name name.c
- From here I was ready to call the executable with any given process inside the shell you provided me.
- I do not think it will earn me any extra points because it was easy to do.
- Under the method runcmd()'s switch statement (' ') I added this line of code to execute the commands with their arguments:

```
execvp(ecmd->argv[0], &ecmd->argv[0]);
```

It executes the executable while passing arguemnts to the executable.

(2) **Implementing Redirection**

- For this I counted on the shell's processor to work—it did not disappoint. I once again modified the code in runcmd() instead under ('>'). Here is the code under the switch statement for redirect:

```
int fd; //establishing a variable to hold the open file
fd = open(rcmd->file, O_WRONLY | O_CREAT); //open and
                                             store created
                                             file in fd
dup2(fd, 1); //built in C method that redirects output to file


runcmd(rcmd->cmd); //now run the command process again
close(fd); //close that open file!
```

(3) **Implementing Parallel Processing**

- This, by far, was the trickiest part of this assignment. I tried to take advantage of the parser but I could not get it working, so I decided to stick my solution in main(). Here is that code:

```
char * strptr; //will store a string, aka, buf dissected
 strptr = strtok(buf, "&"); //if there is "&" string up to "&" is
                              stored in strptr
 int j=0; //process counter to generate enough waits
 int k; //loop variable for FOR LOOP to print wait(&r)


 if (fork1()==0) //guarantees multiple processing without too
                   many processes
 runcmd(parsecmd(strptr)); //start on initial process

 while (strptr != NULL) //enter here if there are more “&”
 {
  j++; //for FOR LOOP to print wait(&r)

  strptr = strtok(NULL, "&"); //grab next line after “&”
  if (fork1()==0) {
   runcmd(parsecmd(strptr)); //run it
   j++;
  } else {


  }


 }

for (k=0; k<j; k++){
 wait(&r); //waiting for process to finish
}

wait(&r); //for initial call to runcmd() outside for loop
```