This file show where the sh.c has been modified and how the nonohup is implemented.

```
 case EXEC:
      ecmd = (struct execcmd*)cmd;
      if(ecmd->argv[0] == 0)
         exit();

      // the nonohup will be parsed as a EXEC cmd by the shell
      // if the EXEC is nonohup we will have the following ecmd->argv

      //ecmd->argv[0]="nonohup"
      //ecmd->argv[1]="program", the program to be run by nonohup
      //ecmd->argv[2] and the rest, are the arguments of the program
      //all we need to do is move the program to ecmd->argv[0]
      // and move the arguments to the position starting from ecmd->argv[1]
      // then we start a new process, run the program in the child process and return the parent process
      // here, the parent process won't wait child process finish, this make the program run in the back

      //try to parse nonohup here
      char nonohup[]="nonohup";
      char* s=ecmd->argv[0];
      for (int i=0;*s!=0;i++)
      {
         if (*s!=nonohup[i])
         {
            break;
         }
         s++;
      }
      if (*s==0)//this is a nonnohup cmd
      {


         int argc=0;
         for(;argc<MAXARGS-1;argc++){
            if(ecmd->argv[argc]==0)
            {
               break;//no more arg, cause it points to nothing;
            }else{
               //it does point to something
               ecmd->argv[argc]=ecmd->argv[argc+1];
```

```
            ecmd->eargv[argc]=ecmd->eargv[argc+1];
          }
        }
        ecmd->argv[argc]=0;
        ecmd->eargv[argc]=0;

        if (ecmd->argv[0]==0)
        {
          panic("nonohup following with no cmd");
          exit();
        }
        if(fork1()==0)//fork a new child, and leave it be, no waiting
        {
          close(1);
          open("nonohup.output",O_WRONLY|O_CREATE);//redirect  output  for
standar output to"nonohup.output"
          runcmd(cmd);//run it in new process again
          exit();
        sleep(5);//make sure parent process exit first;
        }
      }else{// normal cmd no nonohup

       exec(ecmd->argv[0], ecmd->argv);
       printf(2, "exec %s failed\n", ecmd->argv[0]);
      }
      break;
```