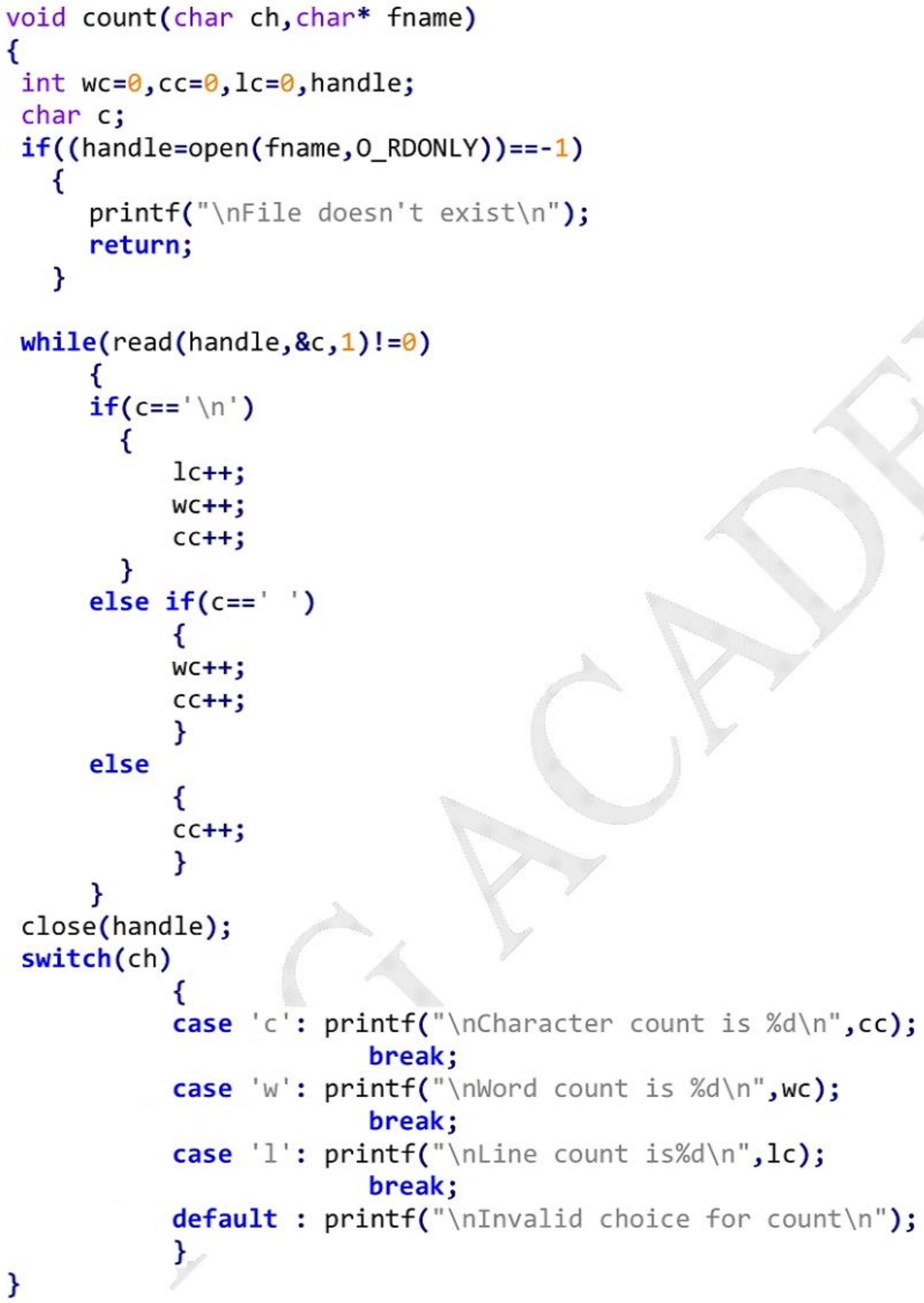
TOYSHELL

```
#include<stdio.h>
#include<fcntl.h>
#include<unistd.h>
#include<string.h>
#include<dirent.h>

void list(char ch,char* dn)
{
 DIR* dir;
 struct dirent* file;
 int cnt=0;

 if((dir=opendir(dn))==NULL)
   {
     printf("\nDirectory not found\n");
     return;
   }

 switch(ch)
       {
           case 'f': while((file=readdir(dir))!=NULL)
                           printf("%s\n",file->d_name);
                      break;

           case 'c': while((file=readdir(dir))!=NULL)
                           cnt++;
                      printf("\nNumber of files is %d\n",cnt);
                      break;

           case 'i': while((file=readdir(dir))!=NULL)
                           printf("%s %d\n",file->d_name,file->d_ino);
                      break;

           default:  printf("\nInvalid choice for list\n");
       }
 closedir(dir);
}
```

```
void typeline(char *s,char* fn)
{
 int handle,n,i=0,cnt=0;
 char c;
 if((handle=open(fn,O_RDONLY))==-1){
     printf("\nFile not found\n");
     return;
 }
 if(s[0]=='a'){
     while(read(handle,&c,1)!=0)
            printf("%c",c);
     close(handle);
     return;
 }
 n=atoi(s);
 if(n>0){
     while(read(handle,&c,1)!=0){
             if(c=='\n')
                 i++;
             if(i==n)
                 break;
             printf("%c",c);
     }
 }else if(n<0){
          while(read(handle,&c,1)!=0)
                if(c=='\n')
                      cnt++;
          lseek(handle,0,SEEK_SET);
          while(read(handle,&c,1)!=0){
                if(c=='\n')
                      i++;
                if(i==cnt+n)
                      break;
          }
          while(read(handle,&c,1)!=0)
                printf("%c",c);
 }else
     printf("Invalid choice for typeline");

 printf("\n");
 close(handle);
}
```

```
void search(char ch,char* s,char* fn)
{
 int i=1,handle,j=0,cnt=0;
 char c,buff[80],*p;
 if((handle=open(fn,O_RDONLY))==-1){
     printf("\nFile not found"); return;
 }
 switch(ch)
     {
      case 'f': while(read(handle,&c,1)!=0)
                      {
                      if(c=='\n')
                        {
                           buff[j]='\0';
                           j=0;
                           if(strstr(buff,s)!=NULL)
                             {
                              printf("%d %s\n",i,buff);
                              break;
                             }
                           i++;
                        }
                      else
                           buff[j++]=c;
                      }
                break;
      case 'a': while(read(handle,&c,1)!=0)
                      {
                       if(c=='\n')
                         {
                           buff[j]='\0';
                           j=0;
                           if(strstr(buff,s)!=NULL)
                             {
                                 printf("%d %s\n",i,buff);
                               //break;
                             }
                           i++;
                         }
                      else
                           buff[j++]=c;
                      }
                break;
     case 'c' : while(read(handle,&c,1)!=0)
                      {
                       if(c=='\n')
                           {
                            buff[j]='\0';
                            j=0;
                            if(strstr(buff,s)!=NULL)
                                 {
                                  p=buff;
                                  while((p=strstr(p,s))!=NULL)
                                      {
                                       cnt++;
                                       p++;
                                      }
                                 }
                                 i++;
                           }
                      else
                           buff[j++]=c;
                      }
                      printf("\nNumber of occurences is %d",cnt);
                break;
      default: printf("\nInvalid choice for search\n");
     }
}
```



```
void count(char ch,char* fname)
{
 int wc=0,cc=0,lc=0,handle;
 char c;
 if((handle=open(fname,O_RDONLY))==-1)
   {
     printf("\nFile doesn't exist\n");
     return;
   }

 while(read(handle,&c,1)!=0)
     {
     if(c=='\n')
       {
           lc++;
           wc++;
           cc++;
       }
     else if(c==' ')
           {
           wc++;
           cc++;
           }
     else
           {
           cc++;
           }
     }
 close(handle);
 switch(ch)
           {
           case 'c': printf("\nCharacter count is %d\n",cc);
                        break;
           case 'w': printf("\nWord count is %d\n",wc);
                        break;
           case 'l': printf("\nLine count is%d\n",lc);
                        break;
           default : printf("\nInvalid choice for count\n");
           }
}
```

```
int main()
{
 char cmd[20],t1[3],t2[3],t3[3],t4[3];
 int n;
 system("clear");
 while(1)
      {
           printf("$$ ");
           fgets(cmd,80,stdin);
           n=sscanf(cmd,"%s%s%s%s",t1,t2,t3,t4);
           switch(n)
                {
                 case 1:if(!fork())
                           {
                            execlp(t1,t1,NULL);  perror(t1);
                           }
                           break;
                 case 2:if(!fork())
                           {
                            execlp(t1,t1,t2,NULL);  perror(t1);
                           }
                           break;
                 case 3:if(strcmp(t1,"count")==0)
                                count(t2[0],t3);
                           else if(strcmp(t1,"typeline")==0)
                                typeline(t2,t3);
                           else if(strcmp(t1,"list")==0)
                                list(t2[0],t3);
                           else if(!fork())
                                {
                                 execlp(t1,t1,t2,t3,NULL);   perror(t1);
                                }
                           break;
                 case 4:if(strcmp(t1,"search")==0)
                                search(t2[0],t3,t4);
                           else if(!fork())
                                {
                                 execlp(t1,t1,t2,t3,t4,NULL); perror(t1);
                                }
                           break;
                }
           }
 return 0;}
```