MAJOR ASSIGNMENT-3

UNIX Systems Programming (CSE 3041)

Designing a `cat` like utility in C

This major assignment is targeted to create a C program to simulate the working of the shell command **`cat`**. The command **`cat`**- *concatenate files and print on the standard output*. **`cat`** concatenate FILE(s), or standard input, to standard output. Go through the **`man`** page for more detailed information about **`cat`** and its options. Some simple experiments with **`cat`** are noted below for your reference to implement these facilities in your code.

Working with `cat`

1. **`$ cat`** ⏎
   Enter characters from standard input (i.e. keyboard). After pressing *enter*, the entered character will be displayed on the standard output (.e. monitor).

2. **`$ cat anExistingFilename`** ⏎
   Displays the content of the file, **`anExistingFilename`**

3. **`$ cat > Filename`** ⏎
   `Output redirection`. Redirects the inputted characters to the file, **`Filename`**. This will create the file, if file is not exist and write onto the file or It will overwrite onto the file, if the file is already exist. After entering the input from the keyboard press CTRL+D. CTRL+D is the end-of-file character.

4. **`$ cat >> Filename`** ⏎
   `Output redirection with append`. Same as above but it will append the newly entered characters onto the file without overwrite.

5. **`$ cat Filename1 Filename2 ... Filenamen`** ⏎
   Displays the content of all files on the standard output one after another.

6. **`$ cat < Filename1`** ⏎
   `Input redirection`. **`cat`** program will take input from the file and display on the standard output.

7. **`$ cat -n Filename(s)`** ⏎
   Display files with line number on the left.

Design your program to meet at least the above steps as **`cat`** supports many more options. The system calls **`dup`**, and **`dup2()`** will help to redirects the file descriptors. The **`main()`** function must be supplied with command-line arguments. The above steps will be the test cases to your code and will be passed from command-line during testing. Your code must be equipped to handle different types of possible error(s), if generated.

Solution

CatUtility.c

```
#include <string.h>
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>

int main(int argc,char *argv[])
{
if(strcmp(argv[0],"concat"))
{
}
else
{
fprintf(stderr,"Usage %s error.\n",argv[0]); exit(0);
if(argc==1)
system("cat"); if(argc==2)
if(execl("/bin/cat","cat",argv[1],NULL)) fprintf(stderr,"cat: no
such file or directory\n");
if(argc==3 && !strcmp(argv[1],"-n")) if(execl("/bin/cat","cat","-
n",argv[2],NULL))
fprintf(stderr,"cat: no such file or directory\n");
if(argc==3 && !strcmp(argv[1],">"))
{
int fd=open(argv[2],O_WRONLY | O_CREAT); if (dup2(fd, 1) == -1)
{
perror("Failed to redirect standard output"); return 1;
}
close(fd);
}
if(argc==3 && !strcmp(argv[1],">>"))
{
int fd=open(argv[2],O_WRONLY | O_CREAT | O_APPEND); if (dup2(fd, 1)
== -1)
{
perror("Failed to redirect standard output"); return 1;
}
close(fd);
}
if(argc==3 && !strcmp(argv[1],"<"))
{
int fd=open(argv[2],O_RDONLY); if (dup2(fd, 1) == -1)
{
perror("Failed to redirect standard output"); return 1;
}
close(fd);
}
if(argc>=3)
if(execvp("cat",argv))
fprintf(stderr,"cat: no such file o directory\n");
}
return 0;
}
```