**Operating System and Design (19CS2106A)**

**Advanced Lab- 3**

**Xv6 design, implementation, and customization.**

**1.rename.c**

#include "types.h"

#include "stat.h"

#include "user.h"

#include "fcntl.h"

#include "fs.h"

int n=0,v=0,o=0;

void

help(){

printf (1,"Usage\n");

printf (1,"rename [OPTION] ekspresi\n");

printf (1,"Options:\n");

printf (1," -s : TIdak rename symlink, tetapi rename target\n");

printf (1," -v : Menunjukan file mana saja yang telah di rename, apabila ada\n");

printf (1," -n : Tidak melakukan perubahan apapun\n");

printf (1," -o : Tidak overwrite file yang telah ada\n");

printf (1," -V : Menunjukkan informasi tentang versi lalu exit\n");

exit();

}

void

prog(){

printf (1,"Rename version 1.00\n");

printf(1,"Dibuat oleh Ferdinand Jason, Nurlita Dhuha, Alvin Tanuwijaya, Bagus Aji Sinto\n");

exit();

}

int

isExist(char \*argv)

{

int err2;

if((err2=open(argv,O\_RDWR))>0) return 1;

else return 0;

}

char\*

strcat(char \*d,char \*s)

{

char \*temp=d;

while(\*d) ++d;

while(\*s) \*d++=\*s++;

\*d=0;

return temp;

}

char \*strncpy(char \*s, const char \*t, int n)

{

int i;

char \*os;

os = s;

for (i = 0; i < n; i++)

{

s[i] = t[i];

}

return os;

}

void rename(char \*argv1, char \*argv2)

{

if(argv1[0]=='.' && argv1[1]=='.') return;

char buf[512];

int fd0, fd1, n;

if ((fd0 = open(argv1, O\_RDONLY)) < 0)

{

printf(2, "rename: cannot open %s\n", argv1);

exit();

}

char temp[512];

strncpy(temp, argv1, strlen(argv1));

if (unlink(argv1) < 0)

{

printf(2, "error renameing %s\n", argv1);

exit();

}

if ((fd1 = open(argv2, O\_CREATE | O\_RDWR)) < 0)

{

printf(2, "rename: cannot open %s\n", argv2);

exit();

}

while ((n = read(fd0, buf, sizeof(buf))) > 0)

{

write(fd1, buf, n);

}

close(fd0);

close(fd1);

}

Void

rename\_rek(char \*from,char \*ext1,char \*ext2)

{

//char buff[1024];

int fd0;

struct dirent de;

struct stat st;

if((fd0=open(from,0))<0)

{

printf(2,"rename: cannot open '%s' No such file or directory\n",from);

exit();

}

if(fstat(fd0,&st)<0)

{

printf(2,"rename: cannot stat '%s' No such file or directory\n",from);

exit();

}

int a;

switch(st.type)

{

case T\_FILE:

{

rename(ext1,ext2);

break;

}

case T\_DIR:

{

while(read(fd0,&de,sizeof(de))==sizeof(de)){

int flag=0;

if(de.inum==0 || de.name[0]=='.') continue;

int idx,b,x;

for(a=0;a<strlen(de.name);a++){

idx=0;

if(de.name[a]==ext1[0]){

for(b=a;b<strlen(de.name);b++){

if(de.name[b]!=ext1[idx]){

break;

}

if(idx==strlen(ext1)-1){

x=b-strlen(ext1)+1;

flag=1;

break;

}

idx++;

}

if(flag) break;

}

}

if(!flag)continue;

char temp[500];

strcpy(temp,de.name);

flag=0;idx=0;

for(a=x;a<x+strlen(ext2);a++){

temp[a]=ext2[idx++];

}

if(o && isExist(temp)) continue;

if(v) printf(1,"%s renamed as %s\n",de.name,temp);

if(!n) rename(de.name,temp);

}

break;

}

}

close(fd0);

}

int main(int argc,char \*argv[]){

//rename 's\.ext1\/.ext2/' namafile1 namafile2 ...

char \*ext1,\*ext2;

ext1=(char\*)malloc(100\*sizeof(char));

ext2=(char\*)malloc(100\*sizeof(char));

int idx=0,a,b;

int com;

if(argv[1][0]!='-') com=1;

else{

com=1;

while(argv[com][0]=='-'){

if(argv[com][1]=='n') n=1;

if(argv[com][1]=='v') v=1;

if(argv[com][1]=='h') help();

if(argv[com][1]=='V') prog();

if(argv[com][1]=='o') o=1;

com++;

}

}

for(a=3;a<strlen(argv[com]);a++){

if(argv[com][a]=='/') break;

ext1[idx++]=argv[com][a];

}

a++;

idx=0;

for(;a<strlen(argv[com]);a++){

if(argv[com][a]=='/') break;

ext2[idx++]=argv[com][a];

}

//printf(1,"%s\n",argv[com]);

if(argv[com+1][0]=='\*'){

rename\_rek(".",ext1,ext2);

}

else{

for(a=2;a<argc;a++){

char \*tmp;

tmp=(char\*)malloc(100\*sizeof(char));

strcpy(tmp,argv[a]);

int len=strlen(ext1);

int len2=strlen(argv[a]);

idx=0;

for(b=len2-len;;b++){

tmp[b]=ext2[idx];

idx++;

if(idx==strlen(ext2)) break;

}

for(;idx<strlen(ext1);idx++){

tmp[++b]=0;

}

rename(argv[a],tmp);

}

}

free(ext1);

free(ext2);

exit();

}

**UNIX system programming**

**SetJump**

#include <stdio.h>

#include <stdlib.h>

#include <setjmp.h>

jmp\_buf env;

int A();

int B();

int C();

int main()

{

int r, a=100;

printf("call setjmp to save environment\n");

if ((r=setjmp(env)) == 0)

{

A();

printf("normal return\n");

}

else

printf("back to main() via long jump, r=%d a=%d\n", r, a);

}

int A()

{ printf("enter A()\n");

B();

printf("exit A()\n");

}

int B()

{ printf("enter B()\n");

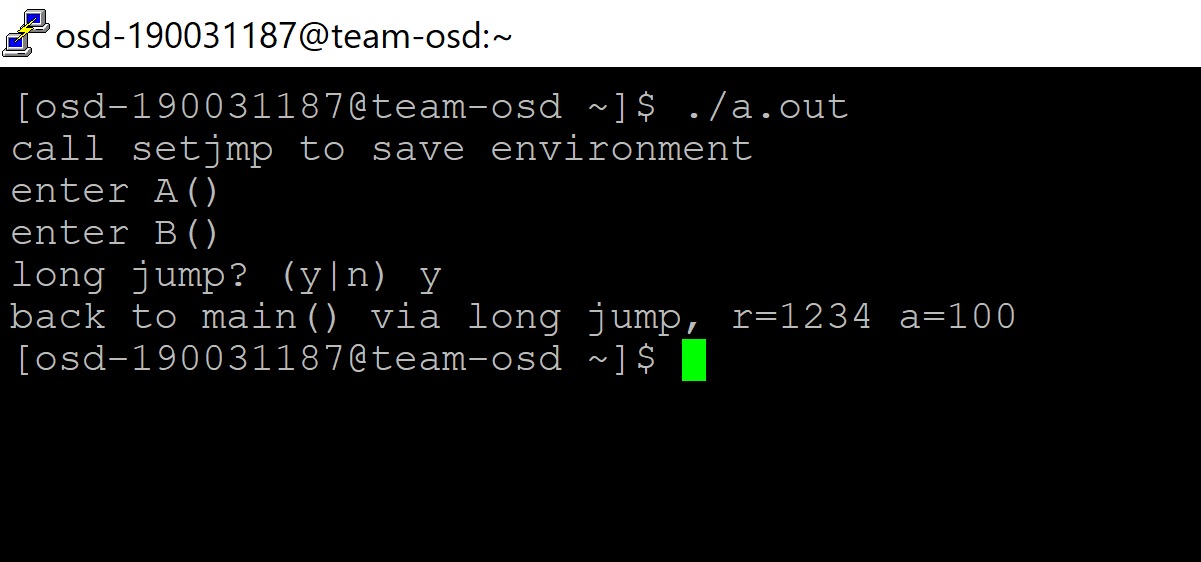
printf("long jump? (y|n) ");

if (getchar()=='y')

longjmp(env, 1234);

printf("exit B()\n");

}

****

**Link C Program with Assembly Code**

simple assembly program code to add two numbers in c program.

.global main

.text

main: # This is called by C library's startup code

mov $message, %rdi # First integer (or pointer) parameter in %rdi

call puts # puts(message)

ret # Return to C library code

message:

.asciz "welcome" # asciz puts a 0 byte at the end

