#################################进程并发

#include<stdio.h>

#include<sys/types.h>

#include<unistd.h>

#include<sys/wait.h>

#include<stdlib.h>

#include<signal.h>

int main()

{

pid\_t pid1, pid2;

pid1=fork();

if (pid1<0)

{

printf("No.1 Error!");

}

else if(pid1 == 0){

printf("This is the No.1 child process");

}

else{

pid2=fork();

if(pid2<0){

printf("No.2 Error!");

}

else if(pid2==0){

printf("This is the No.2 child process");

}

else {

printf("This is the parent process");

}

}

printf("fork end.\n\n");

exit(0);

return 0;

}

#####################################管道通信

#include<unistd.h>

#include<signal.h>

#include<stdlib.h>

#include<stdio.h>

#define MSGSIZE 30

char \*msg1 = "child 1 is sending a message!";

char \*msg2 = "child 2 is sending a message!";

void waiting(int flag)

{

while(flag == 0);

}

int main() {

char inbuf[MSGSIZE];

int p[2];

pid\_t pid1;

pid\_t pid2;

if(pipe(p) == -1) {

perror("pipe call");

exit(0);

}

pid1 = fork();

if(pid1 < 0)

{

printf("Error No.1 fock.\n");

}

else if(pid1 == 0) {

printf("This is 1\n");

close(p[0]);

printf("1 writing\n");

write(p[1], msg1, MSGSIZE);

printf("1 finish.\n\n");

}

else{

pid2 = fork();

if(pid2 < 0) {

printf("Error No.2 fock.\n");

}

else if(pid2 == 0) {

printf("This is 2\n");

close(p[0]);

printf("2 writing\n");

write(p[2], msg2, MSGSIZE);

printf("2 finish.\n");

}

else {

printf("parent p\n");

close(p[1]);

read(p[0], inbuf, MSGSIZE);

printf("%s\n", inbuf);

read(p[0], inbuf, MSGSIZE);

printf("%s\n", inbuf);

}

}

printf("Fock end.\n\n");

exit(0);

return 0;

}

############外加gcc的编译操作，实验报告中已经列出，这里不在给出了。