1.

#include <stdio.h>

#include <unistd.h>

#include <stdlib.h>

#include <pthread.h>

#define n 16

char \*msg = "Hello World";

int count = 0;

pthread\_mutex\_t m = PTHREAD\_MUTEX\_INITIALIZER;

void \*fun(void \*v) {

pthread\_mutex\_lock(&m);

count++;

printf("%d\n", count);

pthread\_mutex\_unlock(&m);

return NULL;

}

int main() {

int pipefd[2], pid;

char buff[1000];

if(pipe(pipefd) < 0) {

exit(1);

}

if(pid = fork() > 0){

write(pipefd[1], msg, n);

close(pipefd[1]);

}

else{

read(pipefd[0], buff, n);

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

}

return 0;

}

2.

#include <stdio.h>

#include <unistd.h>

#include <stdlib.h>

#include <pthread.h>

#define n 16

char \*msg = "Hello World";

int count = 0;

pthread\_mutex\_t m = PTHREAD\_MUTEX\_INITIALIZER;

void \*fun(void \*v) {

pthread\_mutex\_lock(&m);

count++;

printf("%d\n", count);

pthread\_mutex\_unlock(&m);

return NULL;

}

int main() {

int pipefd[2], pid;

char buff[1000];

if(pipe(pipefd) < 0) {

exit(1);

}

if(pid = fork() == 0){

close(pipefd[0]);

write(pipefd[1], msg, n);

}

else{

close(pipefd[1]);

read(pipefd[0], buff, n);

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

}

return 0;

}

3.

#include <stdio.h>

#include <unistd.h>

#include <stdlib.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <fcntl.h>

#include <pthread.h>

#define n 16

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

int pipefd[2], pid;

char buff[1000];

if(pipe(pipefd) < 0) {

exit(1);

}

if(pid = fork() > 0){

write(pipefd[1], argv[1], sizeof(argv[1]));

close(pipefd[0]);

}

else{

close(pipefd[1]);

bzero(buff, sizeof(buff));

read(pipefd[0], buff, sizeof(buff));

int fd = open(buff, O\_RDONLY);

read(fd, buff, sizeof(buff));

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

}

return 0;

}

4.

#include <stdio.h>

#include <unistd.h>

#include <stdlib.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <fcntl.h>

#include <pthread.h>

#define n 16

int count = 0;

pthread\_mutex\_t m = PTHREAD\_MUTEX\_INITIALIZER;

void \*fun(void \*v) {

pthread\_mutex\_lock(&m);

count++;

printf("%d\n", count);

pthread\_mutex\_unlock(&m);

return NULL;

}

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

printf("Before\n");

execl("/bin/ls", "ls", "-l", NULL);

printf("After\n");

return 0;

}

5.

#include <stdio.h>

#include <unistd.h>

#include <stdlib.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <sys/wait.h>

#include <fcntl.h>

#include <pthread.h>

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

pid\_t pid;

if((pid = fork()) > 0){

execl("/bin/ls", "ls", "-l", NULL);

}

else{

wait(NULL);

execl("/bin/pwd", "pwd", NULL);

}

printf("After\n");

return 0;

}

6.

#include <stdio.h>

#include <unistd.h>

#include <stdlib.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <sys/wait.h>

#include <fcntl.h>

#include <pthread.h>

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

pid\_t pid;

int pipefd[2];

pipe(pipefd);

if((pid = fork()) > 0){

dup2(pipefd[1], 1);

close(pipefd[0]);

execl("/bin/ls", "ls", NULL);

}

else{

dup2(pipefd[0], 0);

close(pipefd[1]);

execl("/bin/sort", "sort", NULL);4

}

printf("After\n");

return 0;

}