1. **fork.c**

#include <unistd.h>

main()

{

int pid, status, childpid;

printf("I am the original process with PID %d and PPID %d.\n" ,getpid(),getppid());

pid= fork();

if (pid!=0) /\* parent \*/

{

printf("I am the parent process with PID %d and PPID %d.\n", getpid(),getppid());

childpid= wait(&status);

printf("My child's ppid is %d. It is terminated with exit code %d\n",childpid,status >> 8);

}

else

{ /\* child\*/

printf("I am the child process with PID %d and PPId %d\n",getpid(),getppid());

exit(42);

}

printf("PID %d terminates\n", getpid());

}

1. **echoall.c, execs.c**

//echoall.c

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

{

int i;

char \*\*ptr;

extern char \*\*environ;

for(i=0;i<argc;i++) /\* echo all command-line args \*/

printf("argv[%d]: %s\n", i, argv[i]);

for( ptr=environ; \*ptr !=0 ; ptr++)

printf("%s\n",\*ptr);

exit(0);

}

//execs.c

#include <unistd.h>

#include <sys/types.h>

#include <sys/wait.h>

char \*env\_init[]={ "USER=unknown", "PATH=/tmp", NULL};

int main(void)

{

pid\_t pid;

if ( (pid = fork() ) <0)

printf("Fork error"), exit(1);

else if (pid==0) { /\* specify path name, specify environment \*/

if (execle("/home/USERS/kantarci/SistemProg/process/echoall",

"echoall","myarg1","MYARG2",

(char \*)0, env\_init) <0)

printf("execle error"), exit(1);

}

if (waitpid(pid, NULL, 0) < 0)

printf("Wait error"), exit(1);

getchar();

if ((pid = fork()) <0)

printf("Fork error"), exit(1);

else if (pid == 0) { /\* specify filename, inherit environment \*/

if (execlp("./echoall","echoall","only 1 arg", (char \*) 0) <0)

printf("execlp error"), exit(1);

}

exit(0);

}

1. **data.c**

#include <sys/types.h>

#include <errno.h>

#include <unistd.h>

int glob1 = 6;

int main(void)

{

int var;

pid\_t pid;

var=88;

if ((pid = fork()) <0 )

perror("fork error");

else if (pid == 0) { /\* child \*/

glob1++;

var++;

exit(0);

} else

/\* parent \*/

wait(0);

printf("pid = %d , glob= %d , var= %d\n", getpid(),glob1, var);

exit(0);

}