**Ex.No: 3 UNIX SYSTEM CALLS**

**Date : 12.03.2021**

**Aim**

To write and execute the system calls in unix environment.

**1. Write a C program to invoke the system calls fork(), getpid(), getppid()**

**Algorithm:**

**1.** Start

**2.** Create a file using vi command with filename.c

**3**. Include the header files <stdio.h> for I/O operations, <stdlib.h> to call exit() , <sys/types.h> in order to use pid\_t and <unistd.h> to invoke system calls.

**4**. In main() , declare the variable prgmid of type pid\_t.

**5**. Create a child process using fork() system call.

**6**. Now, there is a parent process and a child process.But, both of them can’t execute simultaneously.

**7**. If the prgmid is equal to 0 , then the child process will execute.

**8**. Then can print the processID and its parent’s processID using getpid(),getppid() respectively.

**9**. Else if the prgmid is greater than 0 , then the parent process will execute and do step 8.

**10**. Otherwise, error statement will be printed.

**11.** Stop.

**Program:**

#include <stdio.h>

#include <sys/types.h>

#include <stdlib.h>

#include <unistd.h>

void main()

{

pid\_t prgmid;

prgmid=fork();

if(prgmid<0)

{

printf("error");

exit(0);

}

else if(prgmid==0)

{

printf("child process");

printf("%d",getpid);

printf("%d",getppid);

}

else

{

printf("parent process");

printf("%d",getpid);

printf("%d",getppid);

}

**}**

**Sample Output:**



**2. Write a C program to invoke the system call wait()**

**Algorithm:**

**1.** Start

**2.** Create a file using vi command with filename.c

**3**. Include the header files <stdio.h> for I/O operations, <stdlib.h> to call exit() , <sys/types.h> in order to use pid\_t, <std/wait.h> to invoke wait() system call and <unistd.h> to invoke system calls.

**4**. Using fork(), create a child process.

**5**. If the pid is greater than 0, make the parent process to wait unitl the child process gets executed using wait() system call.

**6**. If the pid is equal to 0, execute the child process .i.e., print the odd numbers from 1 to 10.

**7**. After that, execute the parent process. i.e., print the even numbers from 1 to 10.

**8**. If the pid is less than 0, then error statement will be printed.

**9.** Stop.

**Program:**

#include<stdio.h>

#include<stdlib.h>

#include<unistd.h>

#include<sys/types.h>

#include<sys/wait.h>

void main()

{

int i;

pid\_t pid;

pid=fork();

if(pid<0)

{

printf("Creation failed");

exit(-1);

}

else if(pid>0)

{

wait(NULL);

printf("\n Parent starts thus printing even numbers");

for(i=2;i<=10;i+=2)

{

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

}

printf("\n End of parent process");

}

else

{

printf("\n Child starts thus printing odd numbers");

for(i=1;i<=9;i++)

{

if(i%2!=0)

{

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

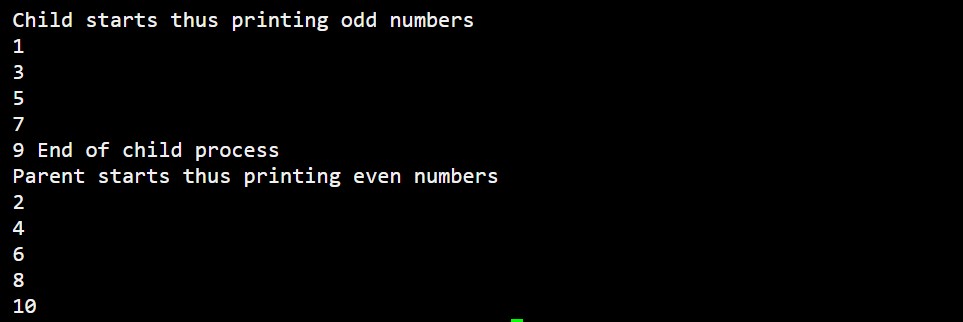
}

}

printf("End of child process");}

}

**Sample Output:**

****

**3. Write a C program to invoke the system call exec()**

**Algorithm:**

**1.** Start

**2.** Create a file using vi command with filename.c

**3**. Include the header files <stdio.h> for I/O operations, <stdlib.h> to call exit() and <unistd.h> to invoke system calls.

**4**. The unix commands can be passed as parameter inside the exec() system call.

**5**. That is, exec() executes the commands inside it and displays the output.

**6**. In this case, we use it ot print the files and directories inside the root directory along with modified time.

**7.** Stop.

**Program:**

#include<stdio.h>

#include<stdlib.h>

#include<unistd.h>

#include<sys/types.h>

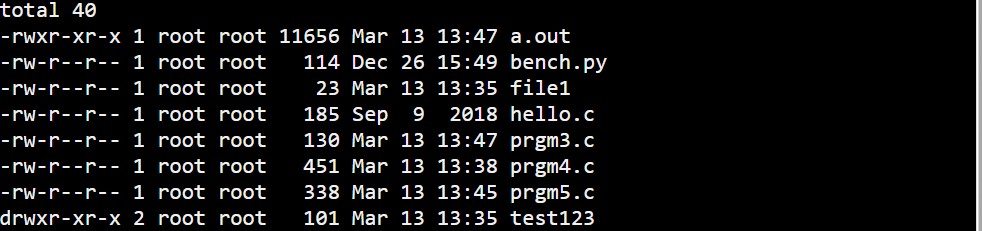
void main()

{

execl("/bin/ls","ls","-l",0,NULL);

}

**Sample Output:**



**4. Write a C program to invoke the system call stat()**

**Algorithm:**

**1.** Start

**2.** Create a file using vi command with filename.c

**3**. Include the header files <stdio.h> for I/O operations, <stdlib.h> to call exit() , <sys/types.h> in order to use pid\_t and <std/stat.h> to invoke stat() system call.

**4.** Using stat() system call, can print the details like uniqueID, groupID, blocksize, no. of blocks and inode number.

**5**. But all these details will be printed only if the specified file exists otherwise it won’t.

**6.** Stop.

**Program:**

#include<stdio.h>

#include<stdlib.h>

#include<time.h>

#include<sys/stat.h>

#include<sys/statvfs.h>

#include<sys/types.h>

void main()

{

struct stat file;

int n;

if((n=stat("file1",&file))==-1)

{

perror("File doesn't exist");

exit(-1);

}

printf("User id: %u \n",file.st\_uid);

printf("Group id: %u \n",file.st\_gid);

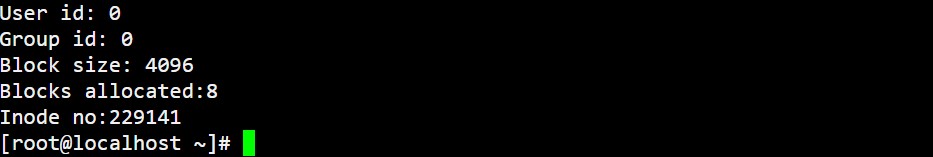
printf("Block size: %ld \n",file.st\_blksize);

printf("Blocks allocated:%ld \n",file.st\_blocks);

printf("Inode no:%ld \n",file.st\_ino);

}

**Sample Output:**

****

**5. Write a C program to invoke the system calls readdir(), opendir()**

**Algorithm:**

**1.** Start

**2.** Create a file using vi command with filename.c

**3**. Include the header files <stdio.h> for I/O operations, <stdlib.h> to call exit() , <sys/types.h> in order to use pid\_t and <dirent.h> to invoke readdir(), opendir() system calls.

**4**. Using opendir(), can open the directory specified.

**5**. Then using readdir(), can read the contents of the opened directory.

**6**. But, the opendir() system call opens the specified directory only if it exists otherwise it shows error.

**7.** Stop.

**Program:**

#include<stdio.h>

#include<stdlib.h>

#include<dirent.h>

#include<sys/types.h>

void main()

{

struct dirent \*dptr;

DIR \*dname;

if((dname=opendir("test123"))==NULL)

{

perror("Directory does not exist");

exit(-1);

}

while(dptr=readdir(dname))

{

printf("Contents of the opened directory: ");

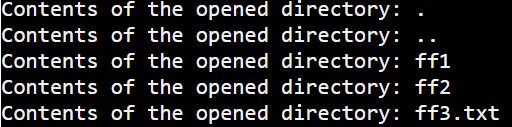
printf("%s\n",dptr->d\_name);

}

closedir(dname);

}

**Sample Output:**

****

|  |  |
| --- | --- |
| **Observation(20)** |  |
| **Record(5)** |  |
| **Total(25)** |  |
| **Initial** |  |

**Result:**

Thus the system calls are invoked and the programs are successfully executed.