Лекция 4

Содержание

- Создание процессов с использованием функций интерфейса POSIX:
  1) в Linux;
  2) в Windows.
- Windows-приложения — функция WinMain.

Создание процессов с помощью семейства системных вызовов exec*.

```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
int main(int argc, char* argv[]){

 fprintf(stdout, "Before child process creating: PARENT ID = %i\n",
                                (int) getpid());

 if(execvp(argv[1], argv)==-1)
      perror("execvp call : ");
 //execvp("ls", argv);

 fprintf(stdout, "Everything is ignored!\n");
 return 0;
}
```

```
int execl(const char *path, const char *arg, ...);

int execlp(const char *file, const char *arg, ...);

int execle(const char *path, const char *arg,..., char * const envp[]);

int execv(const char *path, char *const argv[]);

int execvp(const char *file, char *const argv[]);

int execvpe(const char *file, char *const argv[],char *const envp[]);
```

Output:

```
 ./6ex
Before child process creating: PARENT ID = 5728
1  1.c  2  2.c  2.dat  3  3.c  4  4.c  5  5.c  6  6.c  6ex  6ex.c  test.dat

./6ex  -l *.dat
Before child process creating: PARENT ID = 5741
-rw-r--r-- 1 ewgenij users 3157 2011-02-15 14:34 2.dat
-rw-r--r-- 1 ewgenij users   90 2011-02-15 15:37 test.dat

./6ex  ./5
Before child process creating: PARENT ID = 5923
CHILD:  5923  s=3.14 &s=3669975528
PARENT: 5924  s=2.72 &s=3669975528
```

Совместное использование *fork* и *execvp*:

```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>

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

 pid_t    child_pid;

 child_pid=fork();

 if( child_pid==0)
      execvp("ls", argv);

 fprintf(stdout,"The main program is yet running!\n");
 return 0;
}
```

Родительский процесс продолжает существовать и активен:

```
ewgenij@linux-g5md:~/2011-spring/Lect2> ./7ex -l *.png
The main program is yet running!
-rw-r--r-- 1 ewgenij users 119000 2011-02-15 17:27 exec1.png
```

```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
#include <signal.h>
void oldman(){
 fprintf(stdout, "I'm not yet dead! My ID is %i\n", (int) getpid());
}
void recreation(){
 fprintf(stdout, "Who I am? My ID  is %i\n", (int) getpid());
}
int main(){
 pid_t  child_pid, parent_pid;
int i=0;
 fprintf(stdout, "Before RECREATION %i\n",
                          parent_pid=(int) getpid());
 child_pid=fork();
```

```
 while(i++<5)
  if(child_pid!=0){
    oldman();
    if(i==3)   kill(child_pid,SIGTERM);
  }
  else
    recreation();
 return 0;
}
```

```
I'm not yet dead! My ID is 6526
I'm not yet dead! My ID is 6526
I'm not yet dead! My ID is 6526
Who I am? My ID  is 6527
Who I am? My ID  is 6527
I'm not yet dead! My ID is 6526
I'm not yet dead! My ID is 6526
```

Создание процессов в Windows.
Часть I:использование семейств функций exec* и spawn* в Windows-приложениях.

В заголовочном файле *process.h* содержаться макросы и объявления функций exec*, spawn*, которые могут использоваться для создания процессов. Стандарт *ANSI/ISO C* не включает *process.h,* но этот заголовочный файл и библиотеки времени исполнения, содержащие реализации соответствующих функций присутствуют на многих платформах.

В качестве расширения *process.h* содержится в стандарте *POSIX* (***P****ortable* ***O****perating* ***S****ystem* ***I****nterface for Uni****x***).

Замечание: порты *Cygwin* и *Interix*

Примеры использования exec*:

```
#include <stdio.h>
#include <process.h>

void main(int argc, char* argv[]){
   if(argc<2) return;

    if( _execvp(argv[1], argv)==-1 )
                perror("execvp call : ");
/*
>lab4 calc
>lab4b «cmd» «/c» dir
*/

   printf( "\nProcess was not created." );
   exit( 0 );
  }
```

```
#include <stdio.h>
#include <process.h>

void main(){
  char* argv[]={ "cmd",  "/C",  "dir",  NULL  };

   _execvp(argv[0],argv);

   printf( "\nProcess was not created." );
   exit( 0 );
  }
```

Пример использования spawn*:

```
void main(){
  char* argv[]={ "notepad",  NULL };

   //_spawnvp(_P_OVERLAY, argv[0],argv);
   _spawnvp(_P_NOWAIT, argv[0],argv);

   printf( "\nParent process is yet running." );
   exit( 0 );
  }
```

winspawn.c

```
#include <windows.h>
#include <process.h>

int WINAPI WinMain( HINSTANCE hInstance,
                    HINSTANCE hPrevInstance,
                    LPSTR lpCmdLine, int nCmdShow){
    char* argv[]={ "notepad", NULL };

   //_spawnvp(_P_OVERLAY, argv[0],argv);
    _spawnvp(_P_NOWAIT, argv[0],argv);

   MessageBox(NULL, "Parent process is yet running.“,
                "Message",MB_OK);

   return 0;
 }
```

#define WINAPI __stdcall /*соглашение для вызова функций Win32 API*/

LPSTR	typedef char *LPSTR
HANDLE	typedef PVOID HANDLE
HINSTANCE	typedef HANDLE HINSTANCE

hInstance – дескриптор текущего экземпляра приложения.

hPrevInstance – дескриптор предыдущего экземпляра приложения (рудимент, всегда NULL).

lpCmdLine – параметры командной строки.

nCmdShow – константа, задающая вид окна.