**EXEC FAMILY OF FUNCTIONS**

A parent program can call a exec family of functions to launch a new program. The new program will replace the parent program, but not the process. The new program will get the process ID of the parent program. The segments such as text, initialized data, uninitialized data (bss), and stack of the calling process are overwritten according to the contents of the newly loaded program.

The parent program will not be able to check the status of the exec function on success. Only on error, the parent program can check and take remedial actions. We will discuss six functions, the difference between these functions is just in the format of the parameters passed. We also provided an example for each function.

|  |  |
| --- | --- |
| execl - this function takes the command path, name and optional parameters and NULL parameters.  execl ( "/bin/ls", "ls", "-l", NULL ) | execv – Very identical to execl, except the name and optional parameters and NULL parameters are passed separately in a array  char \*args [ ] = { "ls", "-l", NULL };  execv ("/bin/ls", args ); |
| execle - In addition to execl, this also takes the environment variables in a NULL terminated string array. | execve - In addition to execv, this also takes the environment variables in a NULL terminated string  array. |
| execlp – Same as execl , except it recognizes the path of the command using the $PATH variable. Absolute path is optional | execvp - Same as execv , except it recognizes the path of the command using the $PATH variable  Absolute path is optional |

The functions listed on the left side differ from the right side on one thing: function on the right side take optional parameters of the command in the array of strings. Whereas, you have to list them individually for functions on the left hand side, See the highlighted text in row 1. Functions that end in ‘e’ take the environment variables , and functions that end ‘p’ recognizes the $PATH variables for you to omit the absolute path of the command.

EXECL

1. int execl ( const char \*file, const char \*arg0, ..., NULL);

**file** : is the filename of the file that contains the executable image of the new process.

a**rg0, ..., NULL** : is a variable length list of arguments that are passed to the new process image. Each argument is specified as a null-terminated string, and the list must end with a NULL pointer. The first argument, arg0, is required and must contain the name of the executable file for the new process image. If the new process image is a normal SAS/C main program, the list of arguments will be passed to argv as a pointer to an array of strings.

Example 1:

main ( )

{

execl ( "/usr/bin/cal", "cal", "2017", NULL );

}

another example, echo prints SYSTEM PROGRAMMING

execl ( "/bin/echo", "echo", "SYSTEM PROGRAMMING", NULL );

EXECLE

2. int execle ( const char \*file,

const char \*arg0, ..., NULL,

char \*const envp [ ] ) ;

**file** : is the filename of the program that is to be launched.

**arg0, ..., NULL** : is a variable length list of arguments that are passed to the new program. Each argument is specified as a null-terminated string, and the list must end with a NULL pointer. The first argument, arg0, is required and must contain the name of the executable file for the new process image. The number of strings in the array is passed to the main() function as argc.

**envp** is a pointer to an array of pointers to null-terminated character strings. A NULL pointer is used to mark the end of the array. Each character string pointed to by the array is used to pass an environment variable to the new process image. Each string should have the following form:

" *var* = *value* "

Example 1:

main ( )

{

char \*env[ ] =

{ "SYSTEM PROGRAMMING", NULL };

execle ( "/bin/echo", "echo", env[0], NULL, env );

}

another example

execle ( "/bin/echo", "echo", environ[3], NULL, environ );

EXECLP

3. int execlp(const char \*path, const char \*arg0, ..., NULL);

**path** : identifies the location of the new process in the system. If the path argument contains a slash (/), it is assumed that either an absolute or a relative pathname has been specified. If the path argument does not contain a slash, the directories specified by the PATH environment variable are searched in an attempt to locate the file.

**arg0, ..., NULL** : is a variable length list of arguments that are passed to the function. Each argument is specified as a null-terminated string, and the list must end with a NULL pointer. The first argument, arg0, is required and must contain the name of the executable file.

Example 3:

main ( )

{

execlp ( "ls", "ls", "-lF", NULL );

}

another example

execlp ( "./echo\_1.sh", "./echo\_1.sh", "Jack", "Sam", "Pam", NULL );

EXECV

4. int execv(const char \*file, char \*const argv[] ) ;

**file** : is the filename of the file that contains the executable image of the new process.

**argv** : is a pointer to an array of pointers to null-terminated character strings. A NULL pointer is used to mark the end of the array. Each character string pointed to by the array is used to pass an argument to the new process image. The first argument, argv[0], is required

Example:

main ( )

{

char \*paramList[ ] = { "ls", "-l", NULL} ;

execv ( "/bin/ls", paramList );

}

EXECVE

5. **int execve ( const char \****filename***, char \*const** *argv* **[ ] ,**

**char \*const** *envp* **[ ] ) ;**

**filename** : is the filename of the program to be launched by the function

**argv :** is a pointer to an array of pointers to null-terminated character strings. A NULL pointer is used to mark the end of the array. Each character string pointed to by the array is used to pass an argument to the new process image. The first argument, argv[0], is required and must contain the name of the executable file for the new process image.

**envp** : is a pointer to an array of pointers to null-terminated character strings. A NULL pointer is used to mark the end of the array. Each character string pointed to by the array is used to pass an environment variable to the new process image.

Example :

main ( )

{

char \*paramList[ ] = { "echo", environ[4], NULL };

execve ( "/bin/echo", paramList, environ );

}

EXECVP

6. int execvp ( const char \*path, char \*const argv [ ] ) ;

**path**: identifies the location of the new program. If the path argument contains a slash (/), it is assumed that either an absolute or a relative pathname has been specified. If the path argument does not contain a slash, the directories specified by the PATH environment variable are searched in an attempt to locate the file.

**argv** : is a pointer to an array of pointers to null-terminated character strings. A NULL pointer is used to mark the end of the array. Each character string pointed to by the array is used to pass an argument to the new process image. The first argument, argv[0], is required and must contain the name of the executable file for the new process image.

Example:

main ( )

{

char \*paramList[ ] = { "ls", "-l", NULL} ;

execvp ( "ls", paramList );

}