1. execl() System Function:

In execl() system function takes the path of the executable binary file (i.e. /bin/ls) as the first and second argument. Then, the arguments (i.e. -lh, /home) that you want to pass to the executable followed by NULL. Then execl() system function runs the command and prints the output. If any error occurs, then execl() returns -1. Otherwise, it returns nothing.

Syntax:

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

#include <unistd.h>

int main(void) {

char \*binaryPath = "/bin/ls";

char \*arg1 = "-lh";

char \*arg2 = "/home";

execl(binaryPath, binaryPath, arg1, arg2, NULL);

return 0;

}

1. execlp() System Function:

execl() does not use the PATH environment variable. So, the full path of the executable file is required to run it with execl(). execlp() uses the PATH environment variable. So, if an executable file or command is available in the PATH, then the command or the filename is enough to run it, the full path is not needed.

Syntax:

int execlp(const char \*file, const char \*arg, …, NULL );

#include <unistd.h>

int main(void) {

char \*programName = "ls";

char \*arg1 = "-lh";

char \*arg2 = "/home";

execlp(programName, programName, arg1, arg2, NULL);

return 0;

}

1. execv() System Function:

In execl() function, the parameters of the executable file is passed to the function as different arguments. With execv(), you can pass all the parameters in a NULL terminated array argv. The first element of the array should be the path of the executable file. Otherwise, execv() function works just as execl() function.

Syntax:

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

#include <unistd.h>

int main(void) {

char \*binaryPath = "/bin/ls";

char \*args[] = {binaryPath, "-lh", "/home", NULL};

execv(binaryPath, args);

return 0;

}

1. execvp() System Function:

Works the same way as execv() system function. But, the PATH environment variable is used. So, the full path of the executable file is not required just as in execlp().

Syntax:

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

#include <unistd.h>

int main(void) {

char \*programName = "ls";

char \*args[] = {programName, "-lh", "/home", NULL};

execvp(programName, args);

return 0;

1. execle() System Function:

Works just like execl() but you can provide your own environment variables along with it. The environment variables are passed as an array envp. The last element of the envp array should be NULL. All the other elements contain the key-value pairs as string.

Syntax:

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

#include <unistd.h>

int main(void) {

char \*binaryPath = "/bin/bash";

char \*arg1 = "-c";

char \*arg2 = "echo "Visit $HOSTNAME:$PORT from your browser."";

char \*const env[] = {"HOSTNAME=www.linuxhint.com", "PORT=8080", NULL};

execle(binaryPath, binaryPath,arg1, arg2, NULL, env);

return 0;

}

1. execve() System Function:

Just like execle() you can provide your own environment variables along with execve(). You can also pass arguments as arrays as you did in execv().

Syntax:

int execve(const char \*file, char \*const argv[], char \*const envp[]);

#include <unistd.h>

int main(void) {

char \*binaryPath = "/bin/bash";

char \*const args[] = {binaryPath, "-c", "echo "Visit $HOSTNAME:$PORT

from your browser."", NULL};

char \*const env[] = {"HOSTNAME=www.linuxhint.com", "PORT=8080", NULL};

execve(binaryPath, args, env);

return 0;

}