1. fork()

The fork() command makes a complete copy of the running process and the only way to differentiate the two is by looking at the returned value:

- fork() returns the process identifier (pid) of the child process in the parent, and
- fork() returns 0 in the child.

2. getpid()

Every process can query its own process identifier using the getpid().

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

int main ()
{
  pid_t child_pid;

  printf ("the main program process id is %d\n", (int) getpid ());

  child_pid = fork ();
  if (child_pid != 0) {
    printf ("this is the parent process, with id %d\n", (int) getpid ());
    printf ("the child's process id is %d\n", (int) child_pid);
  }
  else
    printf ("this is the child process, with id %d\n", (int) getpid ());

  return 0;
}
```

3. The exec Family of Functions

There is a family of exec() functions, all of which have slightly different characteristics:

The *char *const argv[]* argument is an array of pointers to null-terminated strings that represent the argument list available to the new program. The first argument, by convention, should point to the filename associated with the file being executed. The array of pointers *must* be terminated by a null pointer.

I) execv()

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

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.

```
#include <unistd.h>

int main(void) {
  char *binaryPath = "/bin/ls";
  char *args[] = {binaryPath, "-lh", "/home", NULL};

  execv(binaryPath, args);

  return 0;
}
```

II) execvp()

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

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().

```
#include <unistd.h>

int main(void) {
  char *programName = "ls";
  char *args[] = {programName, "-lh", "/home", NULL};

  execvp(programName, args);

  return 0;
}
```