As an alternative example, we next consider process creation in Windows. Processes are created in the Win32 API using the CreateProcess() function, which is similar to fork() in that a parent creates a new child process. However, whereas fork() has the child process inheriting the address space of its parent, CreateProcess() requires loading a specified program into the address space of the child process at process creation. Furthermore, whereas fork() is passed no parameters, CreateProcess() expects no fewer than ten parameters.

The C program shown in Figure 3.12 illustrates the CreateProcess() function, which creates a child process that loads the application mspaint.exe. We opt for many of the default values of the ten parameters passed to CreateProcess(). Readers interested in pursuing the details on process creation and management in the Win32 API are encouraged to consult the bibliographical notes at the end of this chapter.

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

int main(VOID)
{
STARTUPINFO si;
PROCESS_INFORMATION pi;

   // allocate memory
   ZeroMemory(&si, sizeof(si));
   si.cb = sizeof(si);
   ZeroMemory(&pi, sizeof(pi));

   // create child process
   if (!CreateProcess(NULL, // use command line
    "C:\\WINDOWS\\system32\\mspaint.exe", // command line
    NULL, // don't inherit process handle
    NULL, // don't inherit thread handle
    FALSE, // disable handle inheritance
    0, // no creation flags
    NULL, // use parent's environment block
    NULL, // use parent's existing directory
    &si,
    &pi))
   {
     fprintf(stderr, "Create Process Failed");
     return -1;
   }
   // parent will wait for the child to complete
   WaitForSingleObject(pi.hProcess, INFINITE);
   printf("Child Complete");

   // close handles
   CloseHandle(pi.hProcess);
   CloseHandle(pi.hThread);
}
```

Figure 3.12 Creating a separate process using the Win32 API.

Two parameters passed to CreateProcess() are instances of the STARTUPINFO and PROCESS_INFORMATION structures. STARTUPINFO specifies many properties of the new process, such as window size and appearance and handles to standard input and output files. The PROCESS_INFORMATION structure contains a handle and the identifiers to the newly created process and its thread. We invoke the ZeroMemory() function to allocate memory for each of these structures before proceeding with CreateProcess().

The first two parameters passed to CreateProcess() are the application name and command line parameters. If the application name is NULL (which in this case it is), the command line parameter specifies the application to load. In this instance we are loading the Microsoft Windows *mspaint.exe* application. Beyond these two initial parameters, we use the default parameters for inheriting process and thread handles as well as specifying no creation flags. We also use the parent's existing environment block and starting directory. Last, we provide two pointers to the STARTUPINFO and PROCESS_INFORMATION structures created at the beginning of the program. In Figure 3.10, the parent process waits for the child to complete by invoking the wait() system call. The equivalent of this in Win32 is WaitForSingleObject(), which is passed a handle of the child process—pi.hProcess— that it is waiting for to complete. Once the child process exits, control returns from the WaitForSingleObject()

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

int main()
{
pid_t pid;

    /* fork a child process */
    pid = fork();

    if (pid < 0) { /* error occurred */
      fprintf(stderr, "Fork Failed");
      exit(-1);
    }
    else if (pid == 0) { /* child process */
      execlp("/bin/ls","ls",NULL);
    }
    else { /* parent process */
      /* parent will wait for the child to complete */
      wait(NULL);
      printf("Child Complete");
      exit(0);
    }
}
```

Figure 3.10 C program forking a separate process.

of the child process at process creation. Furthermore, whereas `fork()` is passed no parameters, `CreateProcess()` expects no fewer than ten parameters.

The C program shown in Figure 3.12 illustrates the `CreateProcess()` function, which creates a child process that loads the application `mspaint.exe`. We opt for many of the default values of the ten parameters passed to `CreateProcess()`. Readers interested in pursuing the details on process creation and management in the Win32 API are encouraged to consult the bibliographical notes at the end of this chapter.

Two parameters passed to `CreateProcess()` are instances of the STARTUPINFO and PROCESS_INFORMATION structures. STARTUPINFO specifies many

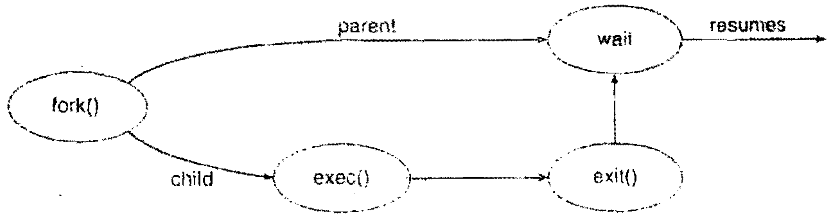

Figure 3.11 Process creation.