An example using fork, execvp and wait

This function could by used by a Unix shell to run a command and wait for the command to finish before going on. It returns the termination status of the command.

It uses function parsecmd(cmd,argv), which is not writtten here, but which breaks cmd at spaces and stores the pieces into argv, followed by a null pointer. For example, parsecmd("eat the banana", argv) will set argv as follows.

```
   argv[0] = "eat"
   argv[1] = "the"
   argv[2] = "banana"
   argv[3] = NULL
```

This example also presumes that there might be other child processes running in background, and that they might terminate while the shell is waiting for the current command to stop. A function called process_terminated is use to handle the termination of a background process. It is not written here.

```
int runcmd(char *cmd)
{
  char* argv[MAX_ARGS];
  pid_t child_pid;
  int child_status;

  parsecmd(cmd,argv);
  child_pid = fork();
  if(child_pid == 0) {
    /* This is done by the child process. */

    execvp(argv[0], argv);

    /* If execvp returns, it must have failed. */

    printf("Unknown command\n");
    exit(0);
  }
  else {
     /* This is run by the parent.  Wait for the child
        to terminate. */

     do {
       pid_t tpid = wait(&child_status);
       if(tpid != child_pid) process_terminated(tpid);
     } while(tpid != child_pid);

     return child_status;
  }
}
```