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

int main (int argc, char*argv[])
{
  pid_t pid; /* Declare a pid type. */
  pid_t mypid; /* Declare a second pid type */
  char cmd[255];
  char prev[255];
  char sb[2];
  int keepGoing = 1;

  mypid = getpid(); /* Get my pid from the OS. */
  printf("The shell, wws shell, is executing.  My PID is: %d. (%x)\n", mypid, mypid);
  /* Print out my process ID data. */

  while ((keepGoing != 0))
  {
    printf("\nWWS->");

    memset (&cmd[0], 0, sizeof(cmd) );
    fgets(cmd, sizeof(cmd), stdin);

        if (strncmp(&cmd[0], "p", 1)==0)
        {
          strcpy(&cmd[0], &prev[0]);
    }
        else if (strncmp(&cmd[0], "rm", 2)==0)
        {
          printf("Are you really sure you want to do this: %s\n", cmd);
          fgets(sb, 2, stdin);

          if (strncmp(&sb[0], "Y", 1)==0)
          {
            // DOIT
          }
          else
          {
            strcpy(cmd, "echo Not deleting it.");
          }
        }


    // Determine what to do.  If the command is exit, exit the shell.
    if (strncmp(&cmd[0], "exit", 4)==0)
    {
      // Exit the shell.
      keepGoing = 0;
      printf("Exiting");
    }
    else
```

```
54      {
55        strcpy(&prev[0], &cmd[0]);
56        // Fork a new process.
57        pid = fork();   /* Fork a child process of this process. */
58
59        if (pid < 0)
60        { /* An error occurred. */ }
61        else if (pid > 0)
62          { /* I am the parent process and my child is the following process. */
63          printf("Start %s as pid is %x. \n", cmd, pid);
64
65          // Determine if the command should execute in the background.
66          if (cmd[0]=='&')
67          {
68              // Execute the program in the background.
69          }
70          else
71          {
72              // Wait for the command to finish.
73              wait(pid);
74          }
75        }
76        else
77        { /* I am the child process. */
78          if (cmd[0]=='&')
79          {
80            system(&cmd[1]);
81          }
82          else
83          {
84            system(&cmd[0]);
85          }
86          exit(0);
87        }
88      }
89    }
90    exit(0);  /* Exit the program, returning 0 as an error code.*/
91  }
92
```