Command Line Arguments in Ada95

Mark Allen Weiss
School of Computer Science
Florida International University
University Park
Miami, FL 33199

When we run a program such as `wc`, we can invoke it with arguments. For instance, the command

```
wc -w echo_args.adb
```

consists of a command name which is the string `"wc"` and two (additional) *arguments*: `"-w"` and `"echo_args.adb"`. These are known as *command line arguments*.

Command line arguments provide an alternate, and sometimes more convenient form for providing input to a program. For instance, under UNIX, in the command

```
wc -w *.adb
```

the sequence `*.adb` expands automatically to match several files, *prior* to calling `wc`.

Consequently, we'd like to have a mechanism by which the arguments on a command line can be accessed in an Ada program. In Ada83 there was no standard way of doing this. Ada95 provides a standard mechanism using the package `Ada.Command_Line`.

The relevant part of the specification for package `Ada.Command_Line` is shown in Figure 1. The use of this package is inherintly implementation dependent. Some systems do not support command line arguments.

As an example of how `Ada.Command_Line` is used, we write a simple program to print out the command line arguments, one per line. This is shown in Figure 2. When you compile this program, run it with a single command line argument * and see what the output is.

```
package Ada.Command_Line is
        -- Return the command name
    function Command_Name return String;

        -- Return number of (additional) arguments
    function Argument_Count return Natural;

        -- Return a particular argument
    function Argument( Number : Positive ) return String;

        -- Routine for returning error codes
    type Exit_Status;

    procedure Set_Exit_Status( Code : Exit_Status );
end Ada.Command_Line;
```

Figure 1: Part of the specification for command line interface

```
with Ada.Text_IO; use Ada.Text_IO;
with Ada.Command_Line; use Ada.Command_Line;

procedure Echo_Args is
begin
        -- Print name of command
    Put_Line( "Command name: " & Command_Name );

        -- Print command line arguments
    for i in 1..Argument_Count loop
        Put_Line( Argument( i ) );
    end loop;

        -- Return status (Success is a constant defined in Ada.Command_Line)
    Set_Exit_Status( Success );
end Echo_Args;
```

Figure 2: Ada95 implementation of an echo command