Question 1B

Program to get and set environtment variables using system calls.

OUTPUT - Environment variables using getenv() and variables after setenv()

Solution

```c
#include <stdio.h>
#include <stdlib.h>

int main() {
    /* Display environment variables using getenv call */
    printf("USER: %s\n", getenv("USER"));
    printf("HOME: %s\n", getenv("HOME"));
    printf("HOST: %s\n", getenv("HOST"));
    printf("ARCH: %s\n", getenv("ARCH"));
    printf("DISPLAY: %s\n", getenv("DISPLAY"));
    printf("PRINTER: %s\n", getenv("PRINTER"));
    printf("PATH: %s\n", getenv("PATH"));

    /* Set two new environment variables */
    const char* environment_var1 = "TEAM1_VAR1";
    const char* environment_val1 = "Team1 1st variable";
    const char* environment_var2 = "TEAM1_VAR2";
    const char* environment_val2 = "Team1 2nd variable";

    /* If the environment variable 1 is not set */
    if (setenv(environment_var1, environment_val1, 1) != 0) {
        perror("setenv");
        return 1;
    }

    /* If the environment variable 2 is not set */
    if (setenv(environment_var2, environment_val2, 1) != 0) {
        perror("setenv");
        return 1;
    }

    /* Printing the environment variables when they are set */
    printf("%s: %s\n", environment_var1, getenv(environment_var1));
    printf("%s: %s\n", environment_var2, getenv(environment_var2));

    return 0;
}
```

Explanation

Display Environment Variables:

```c
    /* Display environment variables using getenv call */
    printf("USER: %s\n", getenv("USER"));
    printf("HOME: %s\n", getenv("HOME"));
    printf("HOST: %s\n", getenv("HOST"));
    printf("ARCH: %s\n", getenv("ARCH"));
    printf("DISPLAY: %s\n", getenv("DISPLAY"));
    printf("PRINTER: %s\n", getenv("PRINTER"));
    printf("PATH: %s\n", getenv("PATH"));
```

- In this section, the code uses the `getenv()` function to retrieve and display the values of several environment variables (USER, HOME, HOST, ARCH, DISPLAY, PRINTER, and PATH).
- `getenv()` takes the *name of an environment variable* as an argument and returns a *pointer* to its value.
  - If an environment variable doesn't exist, `getenv()` returns a `NULL` pointer.

Setting New Environment Variables

```c
    /* Set two new environment variables */
    const char* environment_var1 = "TEAM1_VAR1";
    const char* environment_val1 = "Team1 1st variable";
    const char* environment_var2 = "TEAM1_VAR2";
    const char* environment_val2 = "Team1 2nd variable";
```

In this section, the code defines two new environment variables (`TEAM1_VAR1` and `TEAM1_VAR2`) and their corresponding values.

Set Environment Variables Using `setenv()` :

```c
    /* If the environment variable 1 is not set */
    if (setenv(environment_var1, environment_val1, 1) != 0) {
        perror("setenv");
        return 1;
    }

    /* If the environment variable 2 is not set */
    if (setenv(environment_var2, environment_val2, 1) != 0) {
        perror("setenv");
        return 1;
    }
```

- The code uses the `setenv()` function to set the values of the new environment variables. The `setenv()` function is used to set or update environment variables in a C program. It takes three parameters:
  - **`Name (const char name)`** :
    - This parameter specifies the name of the environment variable you want to set or update.
    - It is a string containing the name of the variable. For example, "PATH" or "MY_VARIABLE".

- **`Value (const char value)`**:
    - This parameter specifies the value to assign to the environment variable identified by the name parameter.
    - It is a string containing the value you want to set. For example, "C:\Program Files" or "123".
- **`Overwrite (int overwrite)`**:
    - The `overwrite` parameter is an integer flag that determines what happens if the environment variable specified by `name` already exists.
    - If `overwrite` is set to 1, the function will update the value of the existing environment variable if it already exists.
        - If it doesn't exist, a new environment variable will be created with the specified name and value.
    - If `overwrite` is set to 0, the function will not update an existing environment variable.
        - If the specified name already exists, the function will not make any changes and may return an error.
        - If the variable does not exist, a new environment variable will be created.

- If `setenv()` encounters an error, it returns a non-zero value, and `perror()` is used to print an error message. In this code, it returns an error if the variables already exist.