Programa comandos personalizados para el sistema operativo

Código del proyecto: ICX0_P4

Producto 3. Automatización básica de la administración de un sistema operativo

Alumno: Jesús Real Tovar
Consultor/ra: Óscar Baltà Fabró
Fecha de entrega: 31 de mayo de 2019

1. Función main, entry point de nuestra aplicación.

```
1   /*
2   Función main, entry point de la aplicación de consola.
3   */
4   int main()
5   {
6     printf("%s\n", "
    ↪ ==========================================
    ↪ ");
7     printf("%s\n", "Introduce la ruta del archivo hosts.");
8     const char * path[255];
9     // Obtenemos mediante consola, la ruta del archivo
10    read_string(path, 255);
11    // Mostramos el contenido del archivo por consola
12    show_content_of_file(path);
13
14    printf("%s\n", "
    ↪ ==========================================
    ↪ ");
15    printf("%s\n", "Introduce la ruta del archivo pares.");
16    const char * diff_path[255];
17    // Obtenemos mediante consola, la ruta del archivo
18    read_string(diff_path, 255);
19    // Hacemos la comparación de los dos archivos y copiamos la diferencia
20    diff_two_files(path, diff_path);
21    // Mostramos el contenido del archivo por consola
22    show_content_of_file(path);
23
24    system("pause");
25    return 0;
26  }
```

Desde la función main, como podemos ver, llamamos a las diferentes funciones para realizar las distintas tareas que requerimos realizar sobre los distintos ficheros.

- Pedimos la ruta del archivo hosts
- Mostramos el contenido del archivo hosts
- Pedimos la ruta del archivo con los pares
- Hacemos la comparación de los dos archivos y aplicamos las diferencias al archivo hosts
- Mostramos el contenido del archivo hosts

2. Función para mostrar el contenido de un archivo.

```
/*
Función para mostrar el contenido de un archivo
*/
void show_content_of_file(const char * path) {
  // Abrimos el archivo
  FILE * ptr_file = open_file(path);
  // Si el archivo no existe retornamos
  if (ptr_file == NULL)
  {
    printf(" %s\n", "El archivo no existe.");
    return;
  }
  printf(" %s %s %s\n", "========== Contenido del archivo: ", path, "
↪ ==========");
  char c;
  c = fgetc(ptr_file);
  // Mientras que no lleguemos al final del archivo
  while (c != EOF)
  {
    // Escribimos en consola cada carácter del archivo
    printf(" %c", c);
    c = fgetc(ptr_file);
  }
  printf("\n");
  // Cerramos el archivo
  fclose(ptr_file);
}
```

Esta función está diseñada para mostrar todo el contenido de un archivo por consola, carácter por carácter. Nos sirve para mostrar los cambios realizados en el archivo hosts y verificar que estos están correctos.

3. Función para chequear el contenido de un archivo.

```
/*
Función para chequear el contenido de un archivo
*/
bool check_content_of_file(const char * path, const char * find) {

  // Abrimos el archivo
  FILE * ptr_file = open_file(path);
  // Si el archivo no existe retornamos
  if (ptr_file == NULL)
  {
    printf(" %s\n", "El archivo no existe.");
    return false;
  }

  bool found = false;
  char * line[1024];

  // Por cada linea del archivo
  while (fgets(line, 1024, ptr_file) != NULL) {
    // Comprobamos si contiene la palabra o frase a buscar
    if ((strstr(line, find)) != NULL) {
      // Si la contiene, seteamos como encontrada y rompemos el bucle
      found = true;
      break;
    }
  }

  // Cerramos el archivo
  fclose(ptr_file);

  // Retornamos found
  return found;
}
```

Con esta función podemos verificar que un archivo contiene cierta palabra o conjunto de palabras. Nos es útil para saber si el archivo hosts contiene alguno de los pares que debemos incorporar desde el archivo de pares.

4. Función para comparar dos archivos, agregar las diferencias a un archivo temporal y aplicarlos al archivo principal.

```
/*
Función para comparar dos archivos, agregar las diferencias a un archivo temporal y aplicarlos al archivo principal
*/
void diff_two_files(const char * path, const char * diff_path) {
  // Abrimos el archivo diff
  FILE * ptr_file_diff = open_file(diff_path);
  // Si el archivo no existe retornamos
  if (ptr_file_diff == NULL)
  {
    printf(" %s\n", "El archivo no existe.");
    return;
  }

  // Creamos el archivo temporal
  FILE * ptr_file_tmp = tmpfile();
  // Si el archivo no existe retornamos
  if (ptr_file_tmp == NULL)
  {
    printf(" %s\n", "Error al crear el archivo temporal.");
    return;
  }

  char * line[1024];

  // Por cada linea del archivo diff
  while (fgets(line, 1024, ptr_file_diff) != NULL) {
    // Comprobamos si contiene el contenido de la linea
    if (!check_content_of_file(path, line)) {
      // La añadimos al archivo temporal
      fputs(line, ptr_file_tmp);
    }
  }
  // Posicionamos el puntero al principio
  fseek(ptr_file_tmp, 0, SEEK_SET);

  // Abrimos el archivo
  FILE * ptr_file = open_file(path);
  // Si el archivo no existe retornamos
  if (ptr_file == NULL)
  {
    printf(" %s\n", "El archivo no existe.");
    return;
  }
  // Posicionamos el puntero al final
  fseek(ptr_file, 0, SEEK_END);
  fputs("\n", ptr_file);
  // Por cada linea del archivo
  while (fgets(line, 1024, ptr_file_tmp) != NULL) {
    // La añadimos al archivo
    fputs(line, ptr_file);
  }

  // Cerramos los archivos
  fclose(ptr_file);
  fclose(ptr_file_diff);
  fclose(ptr_file_tmp); //El archivo temporal se borra solo al cerrarlo
}
```

Con esta función realizamos la tarea más importante de la aplicación, incorporar al archivo hosts los pares del archivo de pares. Para esto seguimos una serie de acciones:

- Abrimos el archivo de pares
- Creamos el archivo temporal donde guardaremos los pares que no contenga el archivo hosts
- Comprobamos que el archivo hosts contenga los pares del archivo de pares, los que no contiene los añadimos al archivo temporal
- Abrimos el archivo hosts
- Añadimos al archivo hosts cada par guardado en el archivo temporal
- Cerramos los archivos
- El archivo temporal es automáticamente borrado por el propio sistema ya que usamos la función *tmpfile*

5. Salida