Práctica nº: 5 Fecha: 23/4/20

Autor: Zdravko Dimitrov Arnaudov

Introducción

La practica a desarrollar se basa en el juego del ahorcado. Para ello, dispondremos de un fichero objeto que almacena una estructura con 900 palabras, un segundo fichero que almacena un par de constantes para acceder a ella y un último fichero objeto el cual contiene una función que nos devuelve un número aleatorio. El objetivo es seleccionar una palabra de forma aleatoria y permitir que el usuario intente adivinar la palabra introduciendo caracteres por teclado. De esta forma, realizaremos un programa main que servirá de estructura principal para el videojuego y un fichero adicional con subrutinas que nos ayudará a simplificar y agilizar el programa.

Para realizar esta prueba, escogemos la opción de juego de 6 caracteres.

Estructura del Código

Contamos con un fichero ‘ahorcado.s’ donde se encuentra el código principal y después el que contiene las subrutinas, ‘funciones1.s’. La funcionalidad del primero se enmarca en realizar operaciones ligeras y llamar a subrutinas cuando sea necesario realizar alguna operación más compleja. Las subrutinas de las que disponemos realizan diferentes funciones: mostrar información por pantalla, inicialización, comprobaciones y tareas más elaboradas.

Se ha pretendido sustraer todo el código posible del main para que sea más sencillo entender y programar su estructura, desde secciones de código repetidas a operaciones complejas.

El juego está implementado de tal forma que tengamos dos variables o cadenas con las que trabajar. Una de ellas almacenará la palabra de la estructura de datos y la segunda es la que se muestra como oculta y la que se irá actualizando por pantalla a medida que el usuario acierta las letras. A la hora de comprobar si se ha acertado el caracter o no, se compara la letra escogida por el usuario con la palabra entera ya almacenada. Si hay alguna coincidencia, se comprueba la operación en la misma posición de la palabra oculta, en caso de que la hayamos acertado anteriormente. Si la letra es un nuevo acierto, se modifica la cadena oculta y se actualiza por pantalla, así hasta terminar.

Pseudo-Código

Comenzaremos con el pseudocódigo del main localizado en el fichero 'ahorcado.s'.

```
swi_Imprime introducción;
swi_Imprime menu juego;

while (opcion no permitida){
        opción = swi_lee @gestion de errores
}
if (opcion == SALIR){
        termina;
}
if (opcion == JUGAR)@{
        numAleatorio = RANDOM (#NUM_PALABRAS);
        *PALABRA[] = PALABRAS[numAleatorio];
        *OCULTA[] = &OCULTA;
        inicializa(PALABRA, OCULTA);

        swi_Imprime  turno,  palabra oculta;
        fallos = 0;
        while (fallos != 6){
                caracter = swi_lee
                if (caracter == 'Ñ'){ @caso especial
                        sigue jugando
                }
                if (caracter != Mayúsculas){
                        leer caracter de nuevo; @gestion de errores
                }

                *PALABRA = @PALABRA;
                *OCULTA = @OCULTA;
                comprobar_resultado (CARACTER,PALABRA, OCULTA);
                if (fallo = true){
                        fallos++;
                }
                if (palabraCompleta == true){
                        terminar;
                } else {
                        swi_Imrpime turno, palabra oculta

                }
        }
```

```
        si (fallos == 6){
                swi_Imprime derrota;
        } else{
                swi_Imprime victoria;
        }

        Mostar palabra correcta;
        volver a jugar;
}
```

Seguidamente, continuaremos con los pseudocódigos de las subrutinas en el fichero 'funciones1.s'

@muestra menú con opciones por pantalla

```
menu:
        swi_Imprime opcion1;
        swi_Imprime opcion2:
        swi_Imprime pedir opcion;
        devolvemos al programa
```

@pide al usuario que introduzca letra y actualiza palabra oculta

```
muestra_turno:
        swi_Imprime pedir turno;
        swi_Imprime palabra OCULTA
        devolvemos al programa
```

@valora si el usuario ha acertado una nueva letra o ha fallado

```
comprueba:
        @devuelve 1 si hubo acierto, 0 si hubo fallo
        acierto = false @para saber si
        LETRA = PALABRA[i]; @inicializamos

        while (caracter != finString){
                LETRA= PALABRA[i];
                if (CARACTER == LETRA){
                        LETRA_OCULTA = OCULTA[i];
                        if (CARACTER != LETRA_OCULTA){
                                OCULTA[i] = CARACTER;
                                acierto = true;
                        }
                }
                *PALABRA ++;
                *OCULTA++;
                LETRA = PALABRA[i];
```

```
        }

        if (acierto == true){
                return 1;
        } else {
                return 0;
        }
        devolvemos al programa
```

@Almacena la palabra de la estructura y inicializa la oculta con símbolos

inicializa_palabras:

```
        LETRA = PALABRAS[i];
        while (LETRA 1= finString){
                PALABRA[i] = LETRA @escribimos en variable palabra
                OCULTA[i] = '-' @asignamos signo a variable oculta

                PALABRAS++@siguiente posición de la palabra en estructura
                PALABRA++ @siguiente posición palabra
                OCULTA++@siguiente posición oculta
                LETRA = PALABRAS[i[ @leemos siguiente caracter
        }
        devolvemos al programa
```

@selecciona aleatoriamente palabra de la estructura y devuelve dirección

selecciona_palabra:

```
        *dir = PALABRAS
        numAleatorio = RANDOM (NUM_PALABRAS);
        desplazamiento = numAleatorio*OFF_PALABRA

        return dir + desplazamiento @direccion de la palabra en la estrucuta
        devolvemos al programa
```

@comprueba si la palabra está adivinada o no.

estado:

```
        signo_oculto = false;
        LETRA = OCULTA[i]; @inicializamos
        while (LETRA != finString){
                if (LETRA == '-'){
                        signo_oculto = true;
                        termina;
                }
                OCULTA++;
                LETRA = OCULTA[i] @nueva letra
        }
        if (signo_oculto == true){
```

return 0; @la palabra no esta completa
} else{
return 1; @la palabra oculta esta adivinada completamente
}
devolvemos al programa

Observaciones

Una vez estructurado el pseudocódigo del main, se comenzó a programar el código y añadiendo nuevas subrutinas a medida que las precisábamos a lo largo de la práctica.
De esta forma, terminamos con un prototipo (bastante más extenso del programa que finalmente tenemos) y probamos su funcionalidad.
A la hora de depurarlo, surgía un error en el momento de calcular el número aleatorio porque devolvía en R0 cero siempre. El problema está en que al disponer de una Raspberry Pi de tercera versión, las direcciones base físicas se encuentran distribuidas de forma distinta, por lo tanto el 'Timer' el cual se usa para la función aleatoria también lo está.
Pienso que el programa convierte la dirección física en una lógica que no es válida y por lo tanto después cuando se accede en modo supervisor a los registros correspondientes, se obtienen valores que no son válidos.
Habiendo averiguado el fallo, cambiamos el fichero 'random' por otro adaptado a la versión de la Raspberry de la que dispongo en este momento y se solucionó.

Más tarde se identificaron ciertos problemas de sintaxis que no es capaz de localizar el compilador pero si el depurador. Algunos de ellos derivaban de usar 'STR' en lugar de 'STRB' a la hora de gestionar las cadenas de caracteres, otros a la hora de gestionar los errores producidos por introducir símbolos no permitidos, etc.

Finalmente teniendo depurado el programa obteniendo una versión funcional y valida comenzamos a optimizar un poco más el código sustituyendo sentencias redundantes por otras más eficaces, creando subrutinas para las secciones de código que se repetían a lo largo del main, mejorando la declaración de datos y otras modificaciones pequeñas.

Por último, comprobamos si la funcionalidad del juego es la esperada introduciendo caracteres incorrectos y buscando las situaciones extremas que el código debería gestionar.

Para ensamblar, hemos usado:

as -o ahorcado.o ahorcado.s
as -o funciones1.o funciones1.s

Para linkar hemos utilizado:

ld -Ttext=18088 -o ahorcado ahorcado.o funciones1.o random.o listadoPalabras6Car.o