Paso 0

a) Capturas de pantalla de la ejecución del aplicativo (con y sin Valgrind).

```
camix@camix-XPS-13-9360:~/Fiuba/Taller/tp0/paso0$ gcc main.c -o tp
camix@camix-XPS-13-9360:~/Fiuba/Taller/tp0/paso0$ ./tp
Hola Mundo
camix@camix-XPS-13-9360:~/Fiuba/Taller/tp0/paso0$ valgrind ./tp
==13419== Memcheck, a memory error detector
==13419== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==13419== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
==13419== Command: ./tp
==13419==
Hola Mundo
==13419==
==13419== HEAP SUMMARY:
==13419==     in use at exit: 0 bytes in 0 blocks
==13419==   total heap usage: 1 allocs, 1 frees, 1,024 bytes allocated
==13419==
==13419== All heap blocks were freed -- no leaks are possible
==13419==
==13419== For counts of detected and suppressed errors, rerun with: -v
==13419== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

b) ¿Para qué sirve Valgrind ? ¿Cuáles son sus opciones más comunes?

Valgrind es una herramienta que sirve para *debuggear* el manejo de la memoria.
Memcheck, la herramienta predeterminada de Valgrind, se encarga de detectar errores en el manejo de la memoria tales como el acceso a memoria que no está reservada, formas incorrectas de liberar la memoria (doble `free`, que no haya ninguno, etc), el uso de valores que no fueron inicializados, y demás.
Massif sirve para saber cuánta memoria de heap usa un programa. También (aunque no es su configuración default) se le puede pedir información sobre la memoria que se ocupa en stack.

c) ¿Qué representa sizeof() ? ¿Cuál sería el valor de salida de sizeof(char) y sizeof(int) ?

La función `sizeof()` recibe el nombre de un tipo de dato y devuelve el tamaño que este ocupa en bytes.
La salida de `sizeof(char)` y `sizeof(int)` generalmente es 1 y 4 respectivamente. En la mayoría de las arquitecturas la memoria que se necesita para almacenar un char es 1 byte, mientras que la memoria que se requiere para almacenar un int es 4 bytes.

d- ¿El sizeof() de una struct de C es igual a la suma del sizeof() de cada uno sus elementos?

No necesariamente. Si bien podría llegar a pasar es muy fácil encontrar algún caso donde, a causa del padding, no se cumpla esta relación.

A continuación presento un ejemplo donde se define una struct en C, se imprime su sizeof() y la suma de los sizeof() de cada uno de sus elementos.

```
#include <stdio.h>

struct alumno {
   char c;
   int i;
}typedef alumno_t;

int main(void) {
   int suma = sizeof(char) + sizeof(int);
   int real = sizeof(alumno_t);
   printf("sizeof() de una struct de C es %d \n", real);
   printf("la suma del sizeof()  de cada uno sus elementos es %d \n",
suma);
   return 0;
}
```

Luego de compilar y ejecutar el código de arriba la salida fue la siguiente:

```
camix@camix-XPS-13-9360:~/Fiuba/Taller/tp0/paso0$ gcc size.c -o size
camix@camix-XPS-13-9360:~/Fiuba/Taller/tp0/paso0$ ./size
sizeof() de una struct de C es 8
la suma del sizeof()  de cada uno sus elementos es 5
```

Con lo cual se concluye que la afirmación "*el sizeof() de una struct de C es igual a la suma del sizeof() de cada uno sus elementos*" es falsa.

e) Investigar la existencia de los archivos estándar: STDIN, STDOUT, STDERR. Explicar brevemente su uso y cómo redirigirlos en caso de ser necesario (caracteres > y <) y como conectar la salida estándar de un proceso a la entrada estándar de otro con un pipe (carácter |).

El STDIN (entrada estándar) es un archivo que se usa para recibir, durante la ejecución de un proceso, datos que van apareciendo con el tiempo y no están estáticos en un archivo. Análogo a esto es el uso del STDOUT (salida estándar) que en lugar de recibir un flujo de datos los emite. Por último, todo proceso tiene un tercer archivo, STDERR, que es destinado a registrar los errores en tiempo de ejecución.

"*STDOUT, generalmente, tiene un búfer asociado para hacerlo más rápido, pero si el programa termina por un error y el búfer no se vacía, entonces se pierde la información. En cambio, STDERR generalmente no tiene un búfer asociado, entonces tenemos la seguridad de que los mensajes se envían siempre y sin retrasos*." [1]

Finalmente a los archivos STDIN, STDOUT Y STDERR se los suele asociar respectivamente a los números 0, 1 y 2.

[1] Apuntes de la cátedra 75.41, segundo cuatrimestre 2018, curso Buchwald, FIUBA.

Al escribir `./mi_programa > archivo` en la consola se está redirigiendo el STDOUT del programa. Es decir, la instrucción que se está dando sería “ejecuta el programa `./mi_programa` usando como salida el archivo `archivo`“. Se obtiene exactamente el mismo resultado al ejecutar el comando `./mi_programa 1 > archivo` ya que en el caso anterior el `1` estaba implícito. En cambio, si la sintaxis es `./mi_programa 2 > archivo` entonces se redirecciona el archivo 2, el STDERR, que pasa a ser el archivo `archivo`. Para redireccionar el archivo de entrada lo más común es ejecutar `./mi_programa < archivo`. Existen múltiples formas de combinar estas sintaxis y redirigir más de un archivo a la vez.

Por otro lado, se puede también redirigir la entrada estándar de un programa hacia la salida de otro. Es decir, el mismo archivo que un programa usa para escribir un stream de datos un segundo programa lo utiliza para ir leyéndolo. La forma de hacerlo por consola es “pipeando” por ejemplo `echo “hola cómo estas” | ./mi-programa` el programa `echo` va a escribir en un archivo de salida estandar “hola mundo” y ese mismo archivo va a ser el de lectura del programa `mi_programa`.

Paso 1

Se hace entrega del archivo “paso1.zip” al sercom.

101055 (Camila Luz Bojman) - 0.1

Comandos Ejecutados

#	Tarea	Comando	Inicio	Duración	Exito?	Observaciones	Diferencias	Archivos Guardados
1	Verificar Normas Codificación	python ./cpplint.py --extensions=h,hpp,c,cpp --filter=`cat filter_options` `find -regextype posix-egrep -regex '.*\.(h|hpp|c|cpp)'`	2019-03-15 15:15:48	0:00:00	No	Se esperaba terminar con un código de retorno 0 pero se obtuvo 1.		Bajar Todo __stdouterr__
1	Compilar C99 simple	make -f Makefile	2019-03-15 15:15:48	0:00:00	No	Se esperaba terminar con un código de retorno 0 pero se obtuvo 2.		Bajar Todo __stdouterr__

Pruebas Realizadas

a) Captura de pantalla mostrando los problemas de estilo detectados. Explicar cada uno.

```
./paso1_wordscounter.c:27:  Missing space before ( in while(  [whitespace/parens] [5]
./paso1_wordscounter.c:41:  Mismatching spaces inside () in if  [whitespace/parens] [5]
./paso1_wordscounter.c:41:  Should have zero or one spaces inside ( and ) in if  [whitespace/parens] [5]
./paso1_wordscounter.c:47:  An else should appear on the same line as the preceding }  [whitespace/newline] [4]
./paso1_wordscounter.c:47:  If an else has a brace on one side, it should have it on both  [readability/braces] [5]
./paso1_wordscounter.c:48:  Missing space before ( in if(  [whitespace/parens] [5]
./paso1_wordscounter.c:53:  Extra space before last semicolon. If this should be an empty statement, use {} instead.
Done processing ./paso1_wordscounter.c
./paso1_main.c:12:  Almost always, snprintf is better than strcpy  [runtime/printf] [4]
./paso1_main.c:15:  An else should appear on the same line as the preceding }  [whitespace/newline] [4]
./paso1_main.c:15:  If an else has a brace on one side, it should have it on both  [readability/braces] [5]
Done processing ./paso1_main.c
./paso1_wordscounter.h:5:  Lines should be <= 80 characters long  [whitespace/line_length] [2]
Done processing ./paso1_wordscounter.h
Total errors found: 11
```

`Missing space before ( in while(  [whitespace/parens]`
Debería haber un espacio en blanco entre la declaración del `while` y la el primer paréntesis que encierra la expresión lógica.

`Mismatching spaces inside () in if  [whitespace/parens]`
La cantidad de espacios en blanco existentes entre la apertura de paréntesis y la primer letra debería ser igual a la cantidad de espacios en blanco existentes entre la última letra dentro del paréntesis y el paréntesis de cierre. Estéticamente esto significa que la expresión lógica debe estar centrada con respecto a los paréntesis que la encierran.

`Should have zero or one spaces inside ( and ) in if  [whitespace/parens]`
La cantidades de espacios blancos mencionados en el error anterior deben ser 0 o 1 únicamente.

`An else should appear on the same line as the preceding }  [whitespace/newline]`
Los `else` deben posicionarse en la misma línea que la llave que cierra el `if` al que hacen referencia.

`If an else has a brace on one side, it should have it on both [readability/braces]`
Si un `else` está precedido por una llave, entonces debería existir otra que la sucede. Lo mismo se da en caso contrario: Si un `else` está sucedido por una llave, entonces debería existir un otra que la antecede.

`Missing space before ( in if(  [whitespace/parens]`
Este error es análogo al primero con la salvedad de que se trata de un `if` en lugar de un `while`.

`Extra space before last semicolon. If this should be an empty statement, use {} instead.  [whitespace/semicolon]`
Hay un espacio en blanco extra entre el el último carácter de la sentencia (final real de la sentencia) y el punto y coma que delimita el fin de la sentencia.

`Almost always, snprintf is better than strcpy  [runtime/printf]`

El error nos está indicando que casi siempre es mejor utilizar la función `snprintf` que `strcopy`.
A veces nos encontramos con varias funciones de la librería estándar que asimilan el comportamiento que buscamos. En estas ocasiones es una buena práctica optar por la función que menos posibilidad de emitir en un error tenga.
En este caso en particular `snprintf` maneja un parámetro para especificar la cantidad de bytes (caracteres) que se desean copiar. Por lo tanto es factible que tenga menos probabilidad de caer en un error de buffer overflow.

`Lines should be <= 80 characters long  [whitespace/line_length]`
El estándar dice que todas las líneas deben ser menores a 80 caracteres. Esto surge de motivos “antiguos” y tal vez obsoletos. No obstante facilita la lectura de cualquier código.

b) Captura de pantalla indicando los errores de generación del ejecutable. Explicar cada uno e indicar si se trata de errores del compilador o del linker.

```
  CC  paso1_main.o
paso1_main.c: In function 'main':
paso1_main.c:22:9: error: unknown type name 'wordscounter_t'
         wordscounter_t counter;
         ^
paso1_main.c:23:9: error: implicit declaration of function 'wordscounter_create' [-Wimplicit-function-declaration]
         wordscounter_create(&counter);
         ^
paso1_main.c:24:9: error: implicit declaration of function 'wordscounter_process' [-Wimplicit-function-declaration]
         wordscounter_process(&counter, input);
         ^
paso1_main.c:25:24: error: implicit declaration of function 'wordscounter_get_words' [-Wimplicit-function-declaration]
         size_t words = wordscounter_get_words(&counter);
                        ^
paso1_main.c:27:9: error: implicit declaration of function 'wordscounter_destroy' [-Wimplicit-function-declaration]
         wordscounter_destroy(&counter);
         ^
<builtin>: recipe for target 'paso1_main.o' failed
make: *** [paso1_main.o] Error 1
```

`unknown type name 'wordscounter_t'`
En la función `main` se declara una variable de tipo `wordscounter_t`. El error se genera porque la estructura `wordscounter_t` no está definida, por lo tanto el compilador no la conoce y no sabe cuánta memoria debe reservar. Este es un error de compilación ya que es el compilador carece de la información suficiente para hacer su tarea.

`implicit declaration of function 'wordscounter_create'`
`implicit declaration of function 'wordscounter_process'`
`implicit declaration of function 'wordscounter_get_words'`
En este caso los errores nos comunican que se llamó a las respectivas funciones pero nunca fueron declaradas previamente. Es decir, el compilador no sabe qué tipo de datos ni cuantos reciben cada una y que tipo de datos devuelven.

c) ¿El sistema reportó algún WARNING? ¿Por qué?
No. El error `implicit declaration of function` citado en el punto anterior en realidad se trata de un warning pero el compilador lo trata como un error porque se le pasó el flag `-Werror`.

Paso 2

Se hace entrega del archivo “paso2.zip” a sercom

Comandos Ejecutados

Tarea	Comando	Inicio	Duración	Exito?	Observaciones	Diferencias	Archivos Guardados
Verificar Normas Codificación	python ./cpplint.py --extensions=h,hpp,c,cpp --filter=`cat filter_options` `find -regextype posix-egrep -regex '.*\.(h|hpp|c|cpp)'`	2019-03-16 19:10:36	0:00:01	Si			Bajar Todo __stdouterr__
Compilar C99 simple	make -f Makefile	2019-03-16 19:10:37	0:00:00	No	Se esperaba terminar con un código de retorno 0 pero se obtuvo 2.		Bajar Todo __stdouterr__

a. Describa en breves palabras las correcciones realizadas respecto de la versión anterior.

Las correcciones que se hicieron fueron mayoritariamente de estilo, especificamente las mencionadas en los errores de la entrega del `paso1.zip`. No obstante, se incluyó en el `main.c` el archivo `wordscounter.h` que contiene la declaración de la estructura y las funciones que el compilador “reclamaba” en el paso anterior.

b. Captura de pantalla indicando la correcta ejecución de verificación de normas de programación.

```
Done processing ./paso2_main.c
Done processing ./paso2_wordscounter.c
Done processing ./paso2_wordscounter.h
Total errors found: 0
```

c. Captura de pantalla indicando los errores de generación del ejecutable. Explicar cada uno e indicar si se trata de errores del compilador o del linker.

```
  CC  paso2_wordscounter.o
In file included from paso2_wordscounter.c:1:0:
paso2_wordscounter.h:7:5: error: unknown type name 'size_t'
     size_t words;
     ^
paso2_wordscounter.h:20:1: error: unknown type name 'size_t'
 size_t wordscounter_get_words(wordscounter_t *self);
 ^
paso2_wordscounter.h:25:49: error: unknown type name 'FILE'
 void wordscounter_process(wordscounter_t *self, FILE *text_file);
                                                 ^
paso2_wordscounter.c:17:8: error: conflicting types for 'wordscounter_get_words'
 size_t wordscounter_get_words(wordscounter_t *self) {
        ^
In file included from paso2_wordscounter.c:1:0:
paso2_wordscounter.h:20:8: note: previous declaration of 'wordscounter_get_words' was here
 size_t wordscounter_get_words(wordscounter_t *self);
        ^
paso2_wordscounter.c: In function 'wordscounter_next_state':
paso2_wordscounter.c:30:25: error: implicit declaration of function 'malloc' [-Wimplicit-function-declaration]
     char* delim_words = malloc(7 * sizeof(char));
                         ^
paso2_wordscounter.c:30:25: error: incompatible implicit declaration of built-in function 'malloc' [-Werror]
paso2_wordscounter.c:30:25: note: include '<stdlib.h>' or provide a declaration of 'malloc'
cc1: all warnings being treated as errors
<builtin>: recipe for target 'paso2_wordscounter.o' failed
make: *** [paso2_wordscounter.o] Error 1
```

```
unknown type name 'size_t'
unknown type name 'FILE'
```

Como se explicó en el **paso 2** respuesta **c)** este es un error de compilación. El código hace referencia a un tipo de datos que no está declarado.

```
paso2_wordscounter.c:17:8: error: conflicting types for
'wordscounter_get_words'
 size_t wordscounter_get_words(wordscounter_t *self) {
        ^
In file included from paso2_wordscounter.c:1:0:
paso2_wordscounter.h:20:8: note: x
 size_t wordscounter_get_words(wordscounter_t *self);
        ^
```

Este error se deriva del anterior. Al no conocer el tipo de datos `size_t` es probable que el compilador haya generado una declaración default de la función que devuelva un entero. Pero, al llegar a la línea donde la función está definida se encuentra con una declaración no concuerda con la anterior.

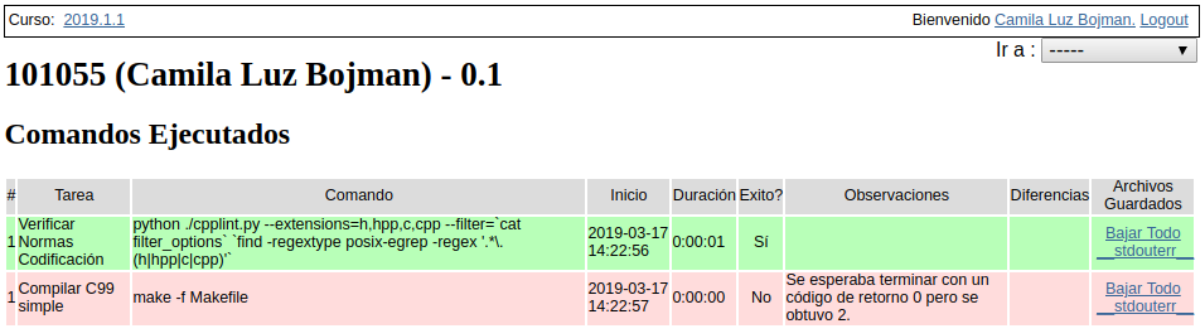
```
paso2_wordscounter.c: In function 'wordscounter_next_state':
paso2_wordscounter.c:30:25: error: implicit declaration of function 'malloc' [-
Wimplicit-function-declaration]
     char* delim_words = malloc(7 * sizeof(char));
                         ^
paso2_wordscounter.c:30:25: error: incompatible implicit declaration of built-
in function 'malloc' [-Werror]
paso2_wordscounter.c:30:25: note: include '<stdlib.h>' or provide a declaration
of 'malloc'
```

Por último, este error se conecta levemente a los anteriores. La función `malloc` se encuentra definida en la misma librería estándar que la estructura `size_t`. Esta librería no está incluida y el compilador no conoce la declaración de la función.

En todos los casos estamos tratando con errores de compilación donde al compilador carece de información.

Paso 3

Se hace entrega del archiv “paso3.zip” a sercom



Curso: 2019.1.1 Bienvenido Camila Luz Bojman. Logout

Ir a : -----

101055 (Camila Luz Bojman) - 0.1

Comandos Ejecutados

#	Tarea	Comando	Inicio	Duración	Exito?	Observaciones	Diferencias	Archivos Guardados
1	Verificar Normas Codificación	python ./cpplint.py --extensions=h,hpp,c,cpp --filter=`cat filter_options` `find -regextype posix-egrep -regex '.*\.(h|hpp|c|cpp)'`	2019-03-17 14:22:56	0:00:01	Sí			Bajar Todo stdouterr
1	Compilar C99 simple	make -f Makefile	2019-03-17 14:22:57	0:00:00	No	Se esperaba terminar con un código de retorno 0 pero se obtuvo 2.		Bajar Todo stdouterr

a. Describa en breves palabras las correcciones realizadas respecto de la versión anterior.

El cambio principal fue la inclusión de librerías estándar. En el archivo `paso3_wordscounter.c` se incluyó la librería mencionada en los errores del paso anterior (`paso3_wordscounter.c`). Y en el código de `paso3_wordscounter.h` se incluyeron las librerías `string.h` y `stdio.h`.

Además, se hicieron los cambios pertinentes en el código a causa de la alteración de los nombres de los archivos.

b. Captura de pantalla indicando los errores de generación del ejecutable. Explicar cada uno e indicar si se trata de errores del compilador o del linker.

```
paso3_main.o: In function `main':
/home/sercom_backend/build/paso3_main.c:27: undefined reference to
`wordscounter_destroy'
```

El error citado nos informa que la función `wordscounter_destroy` no está definida. Este es un error del linker, el compilador conoce la declaración de la función y puede compilar pero el linker desconoce cómo proceder cuando se llama a esta función.

Paso 4

Se hace entrega del archivo “paso4.zip” a sercom

101055 (Camila Luz Bojman) - 0.1

Comandos Ejecutados

#	Tarea	Comando	Inicio	Duración	Exito?	Observaciones	Diferencias	Archivos Guardados
1	Verificar Normas Codificación	python ./cpplint.py --extensions=h,hpp,c,cpp --filter=`cat filter_options` `find -regextype posix-egrep -regex '.*\.(h|hpp|c|cpp)'`	2019-03-17 14:51:17	0:00:00	Si			Bajar Todo stdouterr
1	Compilar C99 simple	make -f Makefile	2019-03-17 14:51:17	0:00:00	Si			Bajar Todo stdouterr

a. Describa en breves palabras las correcciones realizadas respecto de la versión anterior.

Por un lado, al igual que en el `diff` del paso 2 con el 3, se hicieron los reformas necesarias para incluir los archivos con los nombres cambiados.

Además, se agregó una definición de la función que impedía el linkeo del programa. Si bien la función carece de sentencias y no tiene ninguna utilidad permite al linker referenciarla y así solucionar el error que se generaba en el paso 3.

b. Captura de pantalla del resultado de ejecución con Valgrind de la prueba 'TDA'. Describir los errores reportados por Valgrind.

```
==00:00:00:00.000 7193== Memcheck, a memory error detector
==00:00:00:00.000 7193== Copyright (C) 2002-2015, and GNU GPL'd, by Julian Seward et al.
==00:00:00:00.000 7193== Using Valgrind-3.11.0 and LibVEX; rerun with -h for copyright info
==00:00:00:00.000 7193== Command: ./tp input_tda.txt
==00:00:00:00.000 7193== Parent PID: 7192
==00:00:00:00.000 7193==
==00:00:00:00.554 7193==
==00:00:00:00.554 7193== FILE DESCRIPTORS: 3 open at exit.
==00:00:00:00.554 7193== Open file descriptor 2: input_tda.txt
==00:00:00:00.554 7193==    at 0x4113813: __open_nocancel (syscall-template.S:84)
==00:00:00:00.554 7193==    by 0x40A79BF: _IO_file_open (fileops.c:221)
==00:00:00:00.554 7193==    by 0x40A7B40: _IO_file_fopen@@GLIBC_2.1 (fileops.c:328)
==00:00:00:00.554 7193==    by 0x409C2D0: __fopen_internal (iofopen.c:86)
==00:00:00:00.554 7193==    by 0x409C33D: fopen@@GLIBC_2.1 (iofopen.c:97)
==00:00:00:00.554 7193==    by 0x8048517: main (paso4_main.c:14)
==00:00:00:00.554 7193==
==00:00:00:00.554 7193== Open file descriptor 1: /mnt/data/sercom/tmp/prueba.360413.stdout
==00:00:00:00.554 7193==    <inherited from parent>
==00:00:00:00.554 7193==
==00:00:00:00.554 7193== Open file descriptor 0: /home/sercom_backend/test/valgrind.out
==00:00:00:00.554 7193==    <inherited from parent>
==00:00:00:00.554 7193==
==00:00:00:00.554 7193==
==00:00:00:00.554 7193== HEAP SUMMARY:
==00:00:00:00.554 7193==     in use at exit: 1,849 bytes in 216 blocks
==00:00:00:00.554 7193==   total heap usage: 218 allocs, 2 frees, 10,041 bytes allocated
==00:00:00:00.554 7193==
==00:00:00:00.554 7193== 344 bytes in 1 blocks are still reachable in loss record 1 of 2
==00:00:00:00.554 7193==    at 0x402D17C: malloc (in /usr/lib/valgrind/vgpreload_memcheck-x86-linux.so)
==00:00:00:00.554 7193==    by 0x409C279: __fopen_internal (iofopen.c:69)
==00:00:00:00.554 7193==    by 0x409C33D: fopen@@GLIBC_2.1 (iofopen.c:97)
==00:00:00:00.554 7193==    by 0x8048517: main (paso4_main.c:14)
==00:00:00:00.554 7193==
==00:00:00:00.554 7193== 1,505 bytes in 215 blocks are definitely lost in loss record 2 of 2
==00:00:00:00.554 7193==    at 0x402D17C: malloc (in /usr/lib/valgrind/vgpreload_memcheck-x86-linux.so)
==00:00:00:00.554 7193==    by 0x8048685: wordscounter_next_state (paso4_wordscounter.c:35)
==00:00:00:00.554 7193==    by 0x8048755: wordscounter_process (paso4_wordscounter.c:30)
==00:00:00:00.554 7193==    by 0x8048535: main (paso4_main.c:24)
==00:00:00:00.554 7193==
==00:00:00:00.554 7193== LEAK SUMMARY:
==00:00:00:00.554 7193==    definitely lost: 1,505 bytes in 215 blocks
==00:00:00:00.554 7193==    indirectly lost: 0 bytes in 0 blocks
==00:00:00:00.554 7193==      possibly lost: 0 bytes in 0 blocks
==00:00:00:00.554 7193==    still reachable: 344 bytes in 1 blocks
==00:00:00:00.554 7193==         suppressed: 0 bytes in 0 blocks
==00:00:00:00.554 7193==
==00:00:00:00.554 7193== For counts of detected and suppressed errors, rerun with: -v
==00:00:00:00.554 7193== ERROR SUMMARY: 1 errors from 1 contexts (suppressed: 0 from 0)
[SERCOM] Summary
[SERCOM] Command Line: /usr/bin/valgrind --tool=memcheck --trace-children=yes --track-fds=yes --time-stamp=yes --num-callers=20 --error-exitcode=42 --leak-check=full --leak-resolution=med --log-file=valgrind.out --show-reachable=yes --suppressions=suppressions.txt
[SERCOM] Error code configured for Valgrind: 42.
[SERCOM] Valgrind execution result: 42.
[SERCOM] Valgrind result: Failure.
```

```
==00:00:00:00.554 7193== Open file descriptor 2: input_tda.txt
==00:00:00:00.554 7193== 344 bytes in 1 blocks are still reachable in loss
record 1 of 2
==00:00:00:00.554 7193==    at 0x402D17C: malloc (in
/usr/lib/valgrind/vgpreload_memcheck-x86-linux.so)
==00:00:00:00.554 7193==    by 0x409C279: __fopen_internal (iofopen.c:69)
==00:00:00:00.554 7193==    by 0x409C33D: fopen@@GLIBC_2.1 (iofopen.c:97)
==00:00:00:00.554 7193==    by 0x8048517: main (paso4_main.c:14)
```

Estas líneas de error nos muestran que el `main` en la línea 14 hace un llamado a `fopen` pero nunca cierra el archivo. Nos explica que el archivo `input_tda.txt` aún sigue abierto al terminar la ejecución. Y que esta “leak memory” es “still reachable” lo cual significa que no se perdió el puntero a la misma y aún puede liberarse el espacio en memoria allocado.

```
==00:00:00:00.554 7193== 1,505 bytes in 215 blocks are definitely lost in loss
record 2 of 2
==00:00:00:00.554 7193==    at 0x402D17C: malloc (in
/usr/lib/valgrind/vgpreload_memcheck-x86-linux.so)
==00:00:00:00.554 7193==    by 0x8048685: wordscounter_next_state
(paso4_wordscounter.c:35)
==00:00:00:00.554 7193==    by 0x8048755: wordscounter_process
(paso4_wordscounter.c:30)
==00:00:00:00.554 7193==    by 0x8048535: main (paso4_main.c:24)
```

Podemos notar por la descripción del error que se deriva del de un llamado del `main` a `wordscounter_proces` y de este último a `wordscounter_next_state`. Más especificamente en la liena 35 del archivo `paso4_wordscounter.c`. Si vamos a la línea previamente mencionada notamos que se pide memoria para un sting. Esta porción de memoria no es liberada al terminar la ejecución.
La especificación `definitely lost` nos indica que el programa además perdió el puntero a esa porción de memoria.

c. Captura de pantalla del resultado de ejecución con Valgrind de la prueba ‘Long Filename’. Describir los errores reportados por Valgrind.

```
==00:00:00:00.000 7154== Memcheck, a memory error detector
==00:00:00:00.000 7154== Copyright (C) 2002-2015, and GNU GPL'd, by Julian Seward et al.
==00:00:00:00.000 7154== Using Valgrind-3.11.0 and LibVEX; rerun with -h for copyright info
==00:00:00:00.000 7154== Command: ./tp input_extremely_long_filename.txt
==00:00:00:00.000 7154== Parent PID: 7153
==00:00:00:00.000 7154==
**00:00:00:00.523 7154** *** memcpy_chk: buffer overflow detected ***: program terminated
==00:00:00:00.523 7154==    at 0x402FD97: ??? (in /usr/lib/valgrind/vgpreload_memcheck-x86-linux.so)
==00:00:00:00.523 7154==    by 0x40346EB: __memcpy_chk (in /usr/lib/valgrind/vgpreload_memcheck-x86-linux.so)
==00:00:00:00.523 7154==    by 0x804850A: memcpy (string3.h:53)
==00:00:00:00.523 7154==    by 0x804850A: main (paso4_main.c:13)
==00:00:00:00.539 7154==
==00:00:00:00.539 7154== FILE DESCRIPTORS: 2 open at exit.
==00:00:00:00.539 7154== Open file descriptor 1: /mnt/data/sercom/tmp/prueba.360404.stdout
==00:00:00:00.539 7154==    <inherited from parent>
==00:00:00:00.539 7154==
==00:00:00:00.539 7154== Open file descriptor 0: /home/sercom_backend/test/valgrind.out
==00:00:00:00.539 7154==    <inherited from parent>
==00:00:00:00.539 7154==
==00:00:00:00.539 7154==
==00:00:00:00.539 7154== HEAP SUMMARY:
==00:00:00:00.539 7154==     in use at exit: 0 bytes in 0 blocks
==00:00:00:00.539 7154==   total heap usage: 0 allocs, 0 frees, 0 bytes allocated
==00:00:00:00.539 7154==
==00:00:00:00.539 7154== All heap blocks were freed -- no leaks are possible
==00:00:00:00.539 7154==
==00:00:00:00.539 7154== For counts of detected and suppressed errors, rerun with: -v
==00:00:00:00.539 7154== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
[SERCOM] Summary
[SERCOM] Command Line: /usr/bin/valgrind --tool=memcheck --trace-children=yes --track-fds=yes --time-stamp=yes --num-
callers=20 --error-exitcode=42 --leak-check=full --leak-resolution=med --log-file=valgrind.out --show-reachable=yes --
suppressions=suppressions.txt
[SERCOM] Error code configured for Valgrind: 42.
[SERCOM] Valgrind execution result: 1.
[SERCOM] Valgrind result: Success.
```

Las salida de valgrind nos reporta un único error, el acceso de memoria que no está reservada para el buffer que se trata de escribir.

Si vamos a la línea que genera el overflow podemos notar que se usa la función `memcpy` para copiar al buffer `filepath` el contenido del buffer `argv[1]`. El problema es que se le indica que copie todos los caracteres del segundo buffer sin tener en cuenta que el primero no cuenta con tanto espacio.

d. ¿Podría solucionarse este error utilizando la función strncpy ? ¿Qué hubiera ocurrido con la ejecución de la prueba?

`strncpy` recibe un parametro entero n, un buffer de entrada y uno de salida. La función copia hasta n caracteres del buffer de entrada al de salida. Se puede pensar que podría solucionar el error que se nombra en el punto **c)** ya que, en el caso de que n sea mayor a la longitud del buffer de salida, transcribe tantos bytes como entran en el este buffer. Así como utiliza todo el buffer de salida “al máximo” no guarda un byte para para guardar un carácter nulo. Por esto último a la hora de operar con el string de salida se accedería a memoria inválida y se generaría un segmentation fault.

e. Explicar de qué se trata un segmentation fault y un buffer overflow .

El segmentation fault se genera cuando un programa intenta ingresar a una porción de memoria a la que no tiene autorización. O intenta acceder a memoria de la manera en que no tiene permitido, por ejemplo tratando de escribir segmentos de "sólo lectura".

Los programas habitualmente reservan una porción de memoria (buffer) para ser escrita. El buffer overflow se da cuando un programa intenta excederse de la capacidad del buffer al punto que sobrescribiría datos en la memoria adyacente.

Paso 5

Se hace entrega del archivo “paso5.zip” al sercom.

101055 (Camila Luz Bojman) - 0.1

Comandos Ejecutados

#	Tarea	Comando	Inicio	Duración	Exito?	Observaciones	Diferencias	Archivos Guardados
1	Verificar Normas Codificación	python ./cpplint.py --extensions=h,hpp,c,cpp --filter=`cat filter_options` `find -regextype posix-egrep -regex '.*\.(h|hpp|c|cpp)'`	2019-03-18 12:56:56	0:00:00	Sí			Bajar Todo stdouterr
1	Compilar C99 simple	make -f Makefile	2019-03-18 12:56:56	0:00:00	Sí			Bajar Todo stdouterr

Pruebas Realizadas

a. Describa en breves palabras las correcciones realizadas respecto de la versión anterior.

El programa ya no reserva memoria en el main para guardar los parámetros pasados por línea de comando. Tampoco sigue reservando memoria para la cadena de símbolos delimitadores de palabras.
También se agrega un bloque en el main para cerrar archivo en caso de que se haya pasado el parámetro con el nombre del mismo.

b. Describa el motivo por el que fallan las prueba ‘Invalid File’ y ‘Single Word’. ¿Qué información entrega SERCOM para identificar el error? Realice una captura de pantalla.

Invalid file

Descripcion		Archivo inexistente, debe retornar error.					
Comando		./tp invalid_file					
Inicio / Fin		2019-03-18 12:56:57 / 2019-03-18 12:56:58					
#	Tarea	Comando	Duración	Exito?	Observaciones	Diferencias	Archivos Guardados
1	Correr	Prueba normalmente, sin filtros	0:00:00	No	Se esperaba terminar con un código de retorno 1 pero se obtuvo 255.		Bajar Todo __stdout__
2	Valgrind-FailOnError	Correr valgrind a las pruebas fallando si el Valgrind informa error	0:00:01	Sí			Bajar Todo __stdout__ valgrind.out

Sercom nos describe que la respuesta esperada por el programa era un 1. Es decir, que el archivo especificado no exista o no se poseen permisos de lectura. En este caso específico, conociendo la prueba, se le pasa al programa el nombre de un archivo inexistente.

Single Word

Descripcion		Archivo con una única palabra.					
Comando		./tp input_single_word.txt					
Inicio / Fin		2019-03-18 12:56:59 / 2019-03-18 12:56:59					
#	Tarea	Comando	Duración	Exito?	Observaciones	Diferencias	Archivos Guardados
1	Correr	Prueba normalmente, sin filtros	0:00:00	No	La salida estándar no coincide con lo esperado (archivo "__stdout__.diff").	Bajar Ver	Bajar Todo __stdout__
2	Valgrind-FailOnError	Correr valgrind a las pruebas fallando si el Valgrind informa error	0:00:00	No	La salida estándar no coincide con lo esperado (archivo "__stdout__.diff").	Bajar Ver	Bajar Todo __stdout__ valgrind.out

Diferencias

__stdout__.diff

view plain print ?

```
1. --- __stdout__.correcto
2. +++ __stdout__.entregado
3. @@ -1 +1 @@
4. -1
5. +0
```

El error que nos muestra Sercom al procesar el archivo input_single_word.txt es que se esperaba que imprima un 1 (porque el archivo contiene una única palabra) pero se imprimió un 0.

c. Captura de pantalla de la ejecución del comando hexdump . ¿Cuál es el último carácter del archivo input_single_word.txt?

```
camix@camix-XPS-13-9360:~/Fiuba/Taller/7542_fiuba/tp0/paso5$ hd input_single_word.txt
00000000  77 6f 72 64                                       |word|
00000004
camix@camix-XPS-13-9360:~/Fiuba/Taller/7542_fiuba/tp0/paso5$
```

EL último caracter del archivo es una d.

d. Captura de pantalla con el resultado de la ejecución con gdb . Explique brevemente los comandos utilizados en gdb . ¿Por qué motivo el debugger no se detuvo en el breakpoint de la línea 45: self->words++; ?

```
camix@camix-XPS-13-9360:~/Fiuba/Taller/7542_fiuba/tp0/paso5$ gdb ./tp
GNU gdb (Ubuntu 8.1-0ubuntu3) 8.1.0.20180409-git
Copyright (C) 2018 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from ./tp...done.
(gdb) info functions
All defined functions:

File paso5_main.c:
int main(int, char **);

File paso5_wordscounter.c:
void wordscounter_create(wordscounter_t *);
void wordscounter_destroy(wordscounter_t *);
size_t wordscounter_get_words(wordscounter_t *);
void wordscounter_process(wordscounter_t *, FILE *);
static char wordscounter_next_state(wordscounter_t *, char, char);

Non-debugging symbols:
0x0000000000000690  _init
0x00000000000006c0  fclose@plt
0x00000000000006d0  __stack_chk_fail@plt
0x00000000000006e0  strchr@plt
0x00000000000006f0  _IO_getc@plt
0x0000000000000700  __printf_chk@plt
0x0000000000000710  fopen@plt
0x0000000000000720  __cxa_finalize@plt
0x00000000000007e0  _start
0x0000000000000810  deregister_tm_clones
---Type <return> to continue, or q <return> to quit---
0x0000000000000850  register_tm_clones
0x00000000000008a0  __do_global_dtors_aux
0x00000000000008e0  frame_dummy
0x00000000000009f0  __libc_csu_init
0x0000000000000a60  __libc_csu_fini
0x0000000000000a64  _fini
```

El comando `gdb ./pro` arranca a gdb. Es decir, comienza la ejecución del debugger, no del programa.
Luego se ejecutó `info functions` cuya función es enumerar todas las funciones y tipos de datos del programa.

```
(gdb) list wordscounter_next_state
29              int c = getc(text_file);
30              state = wordscounter_next_state(self, state, c);
31          } while (state != STATE_FINISHED);
32      }
33
34      static char wordscounter_next_state(wordscounter_t *self, char state, char c) {
35          const char* delim_words = " ,.;:\n";
36
37          char next_state = state;
38          if (c == EOF) {
```

Luego se usó el comando `list wordscounter_next_state`.

"*Print lines centered around the beginning of function function.*" [2]

Es decir, que el comando imprime una cierta cantidad de líneas previas a la definición de la función que se pase como parámetro y la misma cantidad de líneas desde el comienzo de esta función.

```
(gdb) list
39              next_state = STATE_FINISHED;
40          } else if (state == STATE_WAITING_WORD) {
41              if (strchr(delim_words, c) == NULL)
42                  next_state = STATE_IN_WORD;
43          } else if (state == STATE_IN_WORD) {
44              if (strchr(delim_words, c) != NULL) {
45                  self->words++;
46                  next_state = STATE_WAITING_WORD;
47              }
48          }
```

A continuación se ejecutó el comando `list`. En este caso específico, como las últimas líneas impresas fueron por medio del mismo comando, se imprimen las líneas que le siguen a las últimas impresas.

```
(gdb) break 45
Breakpoint 1 at 0x964: file paso5_wordscounter.c, line 45.
(gdb) run input_single_word.txt
Starting program: /home/camix/Fiuba/Taller/7542_fiuba/tp0/paso5/tp input_single_word.txt
0
[Inferior 1 (process 6367) exited normally]
```

El comando `break 45` ejecutado posteriormente setea un *punto de quiebre* en la línea 45 del *archivo actual*. Siendo el *archivo actual* el último que se imprimió. Y siendo un el *punto de quiebre* una sentencia donde la ejecución se va a detener en caso de pasar por ahí.
Luego se corrió el archivo con el comando `run` y con el nombre del archivo a procesar pasado por parametro `input_single_word.txt`.
Fue evidente que el programa no pasa nunca por la línea 45 porque nunca se detuvo en ella. Es decir, nunca se encontró un delimitador de palabras entonces nunca se sumó uno a

[2] documentación de list function que figura en ftp://ftp.gnu.org/old-gnu/Manuals/gdb/html_node/gdb_46.html

la cantidad de palabras procesadas. Esto explica perfectamente el error que documenta sercom, se esperaba que el programa imprima 1 pero imprimió 0.
Finalmente, terminada la ejecución del programa y el debuggueo se cerro gdb con el comando `quit`.

Paso 6

Se hace entrega del archivo “paso6.zip” al sercom.

#	Tarea	Comando	Inicio	Duración	Exito?	Observaciones	Diferencias	Archivos Guardados
1	Verificar Normas Codificación	python ./cpplint.py --extensions=h,hpp,c,cpp --filter=`cat filter_options` `find -regextype posix-egrep -regex '.*\.(h|hpp|c|cpp)'`	2019-03-18 16:18:13	0:00:00	Sí			Bajar Todo stdouterr
1	Compilar C99 simple	make -f Makefile	2019-03-18 16:18:13	0:00:00	Sí			Bajar Todo stdouterr

a. Describa en breves palabras las correcciones realizadas respecto de la versión anterior.
Se corrigió la salida en caso de error, deja de ser un -1 y pasa a ser un 1 como la consigna lo indica.
También se comenzó a contemplar en el contador de palabras tenga en cuenta a la última palabra del archivo si está no ternía un delimitador al final.
O sea que se corrigieron los errores que se generaba en las pruebas del **paso 5**

b. Captura de pantalla mostrando t odas las entregas realizadas , tanto exitosas como fallidas.

Curso: 2019.1.1 Bienvenido Camila Luz Bojman. Logout
Ir a : -----

Administración de Entregas

Ejercicio	Resultado	Fecha	Duración	Observaciones	Operaciones
0.1 (Contador de Palabras)	Aceptado	2019-03-18 16:18:06	0:00:06		Corrida Bajar Navegar PDF
0.1 (Contador de Palabras)	Rechazado	2019-03-18 12:56:44	0:00:06		Corrida Bajar Navegar PDF
0.1 (Contador de Palabras)	Rechazado	2019-03-17 14:51:11	0:00:06		Corrida Bajar Navegar PDF
0.1 (Contador de Palabras)	Rechazado	2019-03-17 14:22:50	0:00:01		Corrida Bajar Navegar PDF
0.1 (Contador de Palabras)	Rechazado	2019-03-16 19:10:34	0:00:01		Corrida Bajar Navegar PDF
0.1 (Contador de Palabras)	Rechazado	2019-03-15 15:15:36	0:00:01		Corrida Bajar Navegar PDF

c. Captura de pantalla mostrando la ejecución de la prueba ‘Single Word’ de forma local con las distintas variantes indicadas.

```
camix@camix-XPS-13-9360:~/Fiuba/Taller/7542_fiuba/tp0/paso6$ make
  CC  paso6_main.o
  LD  tp
camix@camix-XPS-13-9360:~/Fiuba/Taller/7542_fiuba/tp0/paso6$ ./tp input_single_word.txt
1
camix@camix-XPS-13-9360:~/Fiuba/Taller/7542_fiuba/tp0/paso6$ ./tp <input_single_word.txt
1
camix@camix-XPS-13-9360:~/Fiuba/Taller/7542_fiuba/tp0/paso6$ ./tp <input_single_word.txt >output_single_word.txt
camix@camix-XPS-13-9360:~/Fiuba/Taller/7542_fiuba/tp0/paso6$
```