universidad de león

Escuela de Ingenierías

Evaluación de Código C

Diseño y programación segura

Alumnos:

González González, Ángel

Máster Universitario en Investigación de Ciberseguridad

Índice

1.1- Ejemplo 1

```
#include <stdio.h>

#include <stddef.h>

const char *p;

char *funcion1(void) {
   char array[10] = “Mi Cadena”;
   /* Initialize array */
   return array;
}
void funcion2(void) {
    const char c_str[] = "Todo va bien";
    p = c_str;
}
void funcion3(void){
      printf(“%s\n”,p);
}
int main(void){
      p=funcion1();
      printf(“%s\n”,p);
      funcion2();
      funcion3();
      printf(“%s\n”,p);
      return0;
}
```

1. Define la regla que se incumple y propón una alternativa más adecuada según el SEI CERT C.

En ese código se entá incumpliendo la regla DCL30-C, por dos razones, la primera es la forma en la declaración del array en la funcion1, ya que está incumpliendo dentro de la regla DCL30-C Return Values, puesto que esa función1, retorna un puntero de una variable local, sería más adecuado poner:

```
char *funcion1(char array[]) {


   return;
}
```

```
int main(void){
      char array[10] = “Mi Cadena”;
      p=funcion1(array);
      /…/
}
```

De esta forma ya no estamos pasando una variable local ya que lo declaramos en el main.

Otra regla que se incumple es en DCL30-C, en la función2, se está incumpliendo en DCL30-C el Static Variables, ya que estamos declarando una variable fuera de la función, y estamos asignandole un valor dentro de una función, de esta forma el puntero está apuntando al valor de c_str, que se va a terminar cuando la funcion2 termine. La mejor forma será:

```
void funcion2(void) {
    const char c_str[] = "Todo va bien";
    const char *p = c_str;
}
```

Declarando p dentro de la función y quitando la declaración de fuera, de esta forma p no va a tomar valores indeterminados fuera de la función.

1.2- Ejemplo 2

```
#include <stdlib.h>


struct flexArrayStruct {
  int num;
  int data[1];
};


void func(size_t array_size) {
  /* Space is allocated for the struct */
  struct flexArrayStruct *structP
    = (struct flexArrayStruct *)
     malloc(sizeof(struct flexArrayStruct)
          + sizeof(int) * (array_size - 1));
  if (structP == NULL) {
    /* Handle malloc failure */
  }

  structP->num = array_size;

  /*
   * Access data[] as if it had been allocated
```

```
     * as data[array_size].
     */
    for (size_t i = 0; i < array_size; ++i) {
      structP->data[i] = 1;
    }
  }
```

1. ¿Qué hace el siguiente segmento de código?

Este código realiza la función de crea un array dinámico mediante estructuras

2. De haber algún problema ¿Podrías decir la línea en la que se encuentra?

Hay un error en la declaración del struct, que es la línea 5, ya que tiene declarado como data[1], pero si quiere que sea dinámico al estar poniendo el 1 no va a crecer más con lo cuál solo vas a poder tener un valor.

3. Define la regla que se incumple y propón una alternativa correcta siguiendo el SEI CERT C.

Está incumpliendo la regla DCL38-C, y la solución correcta para realizar su función, es en la declaración del struct flexArrayStruct, en int data[1], poner int data[], de esta forma el array ya va a ser dinámico y va a poder crecer sin salirse de memoria.

1.3- Ejemplo 3

```
#include <stdio.h>

extern void f(int i);

void func(int expr) {
  switch (expr) {
    int i = 4;
    f(i);
  case 0:
    i = 17;
  default:
    printf("%d\n", i);
  }
}
```

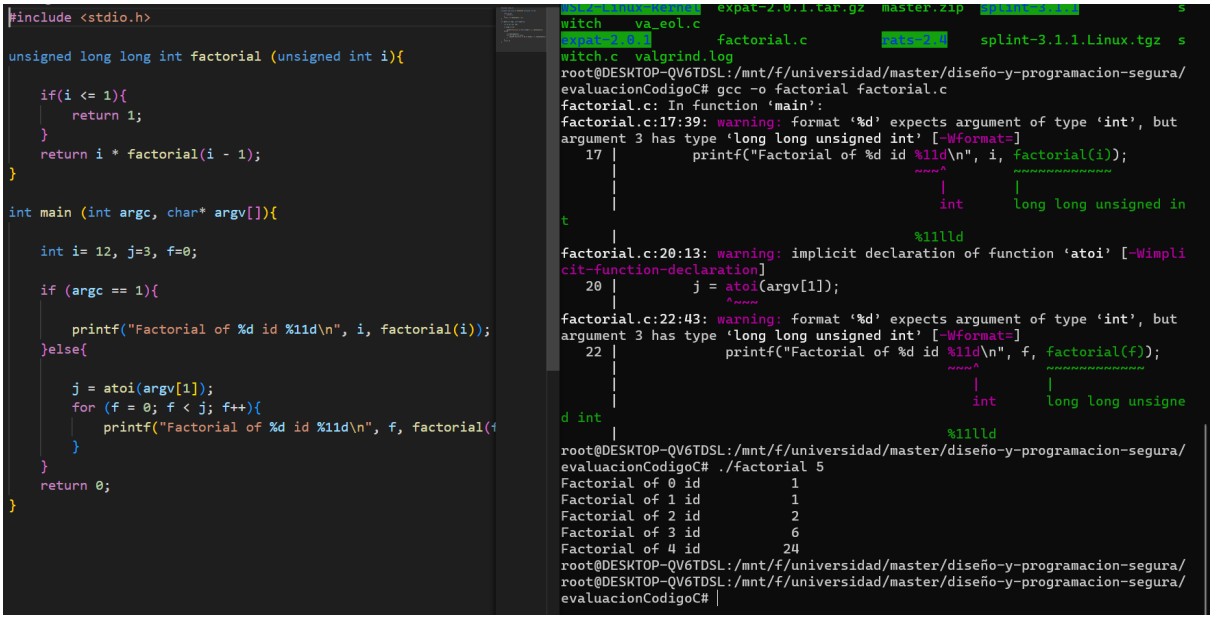
1. ¿Que hace el siguiente segmento de código si invocamos la funció́n func con un 0?

Si se introduce un 0, la salida que va a dar es i = 17, con lo cual la salida es 17.

2. De haber algún problema ¿Podrías decir la línea en la que se encuentra?

La declaración de i y la llamada a la función, se está realizando dentro del switch, se encuentra en la línea 7 y 8. De esta forma se incumple la regla DCL41 Do not declare variables inside a switch before the first case.

3. Crea un fichero con un main y ejecuta el segmento de codigo.



```
#include <stdio.h>

unsigned long long int factorial (unsigned int i){

    if(i <= 1){
        return 1;
    }
    return i * factorial(i - 1);
}

int main (int argc, char* argv[]){

    int i= 12, j=3, f=0;

    if (argc == 1){

        printf("Factorial of %d id %11d\n", i, factorial(i));
    }else{

        j = atoi(argv[1]);
        for (f = 0; f < j; f++){
            printf("Factorial of %d id %11d\n", f, factorial(f
        }
    }
    return 0;
}
```

```
root@DESKTOP-QV6TDSL:/mnt/f/universidad/master/diseño-y-programacion-segura/evaluacionCodigoC# gcc -o factorial factorial.c
factorial.c: In function 'main':
factorial.c:17:39: warning: format '%d' expects argument of type 'int', but argument 3 has type 'long long unsigned int' [-Wformat=]
   17 |         printf("Factorial of %d id %11d\n", i, factorial(i));
      |                                       ~~~^        ~~~~~~~~~~~~
      |                                          |        |
      |                                          int      long long unsigned int
      |                                       %11lld
factorial.c:20:13: warning: implicit declaration of function 'atoi' [-Wimplicit-function-declaration]
   20 |         j = atoi(argv[1]);
      |             ^~~~
factorial.c:22:43: warning: format '%d' expects argument of type 'int', but argument 3 has type 'long long unsigned int' [-Wformat=]
   22 |             printf("Factorial of %d id %11d\n", f, factorial(f));
      |                                           ~~~^        ~~~~~~~~~~~~
      |                                              |        |
      |                                              int      long long unsigned int
      |                                           %11lld
root@DESKTOP-QV6TDSL:/mnt/f/universidad/master/diseño-y-programacion-segura/evaluacionCodigoC# ./factorial 5
Factorial of 0 id           1
Factorial of 1 id           1
Factorial of 2 id           2
Factorial of 3 id           6
Factorial of 4 id          24
root@DESKTOP-QV6TDSL:/mnt/f/universidad/master/diseño-y-programacion-segura/
root@DESKTOP-QV6TDSL:/mnt/f/universidad/master/diseño-y-programacion-segura/evaluacionCodigoC#
```

4. Propón una solución al ejemplo que cumpla con las normal del CMU

Una forma de solucionalo es sacando la declaración de int i = 4, y la llamada de la función para antes del switch:

```
void func(int expr) {
      int i = 4;
      f(i);
      switch (expr) {
            case 0:
```

De esta forma la declaración y la llamada a la función, se realiza fuera del switch.

5. Realiza un analisis estático del código erróneo y copia en tu solución el resultado. Utiliza las herramientas:

(a) rats

```
$ ./rats ../switch.c
Entries in perl database: 33
Entries in ruby database: 46
Entries in python database: 62
Entries in c database: 334
Entries in php database: 55
Analyzing ../switch.c
Total lines analyzed: 24
Total time 0.000747 seconds
32128 lines per second
$
```

(b) cppchecker

```
$ cppcheck switch.c
Checking switch.c ...
$
```

(c) splint

```
$ splint switch.c
Splint 3.1.2 --- 20 Feb 2018

../../dise\377\377o-y-programacion-segura/evaluacionCodigoC/switch.c: (in function f)
../../dise\377\377o-y-programacion-segura/evaluacionCodigoC/switch.c:6:12:
    Parameter i not used
  A function parameter is not used in the body of the function. If the argument
  is needed for type compatibility or future plans, use /*@unused@*/ in the
  argument declaration. (Use -paramuse to inhibit warning)
../../dise\377\377o-y-programacion-segura/evaluacionCodigoC/switch.c: (in function func)
../../dise\377\377o-y-programacion-segura/evaluacionCodigoC/switch.c:14:8:
    Fall through case (no preceding break)
  Execution falls through from the previous case (use /*@fallthrough@*/ to mark
  fallthrough cases). (Use -casebreak to inhibit warning)
../../dise\377\377o-y-programacion-segura/evaluacionCodigoC/switch.c:16:11:
    Fall through case (no preceding break)
../../dise\377\377o-y-programacion-segura/evaluacionCodigoC/switch.c:12:15:
    Statement after switch is not a case: int i = 4
  The first statement after a switch is not a case. (Use -firstcase to inhibit
  warning)
../../dise\377\377o-y-programacion-segura/evaluacionCodigoC/switch.c:4:13:
    Function exported but not used outside switch: f
  A declaration is exported, but not used outside this module. Declaration can
  use static qualifier. (Use -exportlocal to inhibit warning)
   ../../dise\377\377o-y-programacion-segura/evaluacionCodigoC/switch.c:8:2:
   Definition of f
../../dise\377\377o-y-programacion-segura/evaluacionCodigoC/switch.c:10:6:
    Function exported but not used outside switch: func
   ../../dise\377\377o-y-programacion-segura/evaluacionCodigoC/switch.c:19:1:
   Definition of func

Finished checking --- 6 code warnings
$
```

(d) vera++

```
$ vera++ switch.c
switch.c:1: trailing whitespace
switch.c:1: no copyright notice found
switch.c:2: trailing whitespace
switch.c:3: trailing whitespace
switch.c:4: trailing whitespace
switch.c:5: trailing whitespace
switch.c:6: trailing whitespace
switch.c:7: trailing whitespace
switch.c:8: trailing whitespace
switch.c:8: closing curly bracket not in the same line or column
switch.c:9: trailing whitespace
switch.c:10: trailing whitespace
switch.c:11: trailing whitespace
switch.c:12: trailing whitespace
switch.c:13: trailing whitespace
switch.c:14: trailing whitespace
switch.c:15: trailing whitespace
switch.c:16: trailing whitespace
switch.c:17: trailing whitespace
switch.c:18: trailing whitespace
switch.c:18: closing curly bracket not in the same line or column
switch.c:19: trailing whitespace
switch.c:19: closing curly bracket not in the same line or column
switch.c:20: trailing whitespace
switch.c:21: trailing whitespace
switch.c:22: trailing whitespace
switch.c:23: trailing whitespace
switch.c:24: no newline at end of file
switch.c:24: closing curly bracket not in the same line or column
$
```

(e) valgrind

```
--9303-- REDIR: 0x49226a0 (libc.so.6:strnlen) redirected to 0x48311d0 (_vgnU_ifunc_wrapper)
--9303-- REDIR: 0x4922260 (libc.so.6:strcspn) redirected to 0x48311d0 (_vgnU_ifunc_wrapper)
--9303-- REDIR: 0x4923830 (libc.so.6:strncasecmp) redirected to 0x48311d0 (_vgnU_ifunc_wrapper)
--9303-- REDIR: 0x4922200 (libc.so.6:strcpy) redirected to 0x48311d0 (_vgnU_ifunc_wrapper)
--9303-- REDIR: 0x4923980 (libc.so.6:memcpy@@GLIBC_2.14) redirected to 0x48311d0 (_vgnU_ifunc_wrapper)
--9303-- REDIR: 0x4940f80 (libc.so.6:wcsnlen) redirected to 0x48311d0 (_vgnU_ifunc_wrapper)
--9303-- REDIR: 0x493f890 (libc.so.6:wcscpy) redirected to 0x48311d0 (_vgnU_ifunc_wrapper)
--9303-- REDIR: 0x4922850 (libc.so.6:strpbrk) redirected to 0x48311d0 (_vgnU_ifunc_wrapper)
--9303-- REDIR: 0x4922130 (libc.so.6:index) redirected to 0x48311d0 (_vgnU_ifunc_wrapper)
--9303-- REDIR: 0x4922660 (libc.so.6:strlen) redirected to 0x48311d0 (_vgnU_ifunc_wrapper)
--9303-- REDIR: 0x492bbd0 (libc.so.6:memrchr) redirected to 0x48311d0 (_vgnU_ifunc_wrapper)
--9303-- REDIR: 0x4923880 (libc.so.6:strcasecmp_l) redirected to 0x48311d0 (_vgnU_ifunc_wrapper)
--9303-- REDIR: 0x4923400 (libc.so.6:memchr) redirected to 0x48311d0 (_vgnU_ifunc_wrapper)
--9303-- REDIR: 0x493f960 (libc.so.6:wcslen) redirected to 0x48311d0 (_vgnU_ifunc_wrapper)
--9303-- REDIR: 0x4922b10 (libc.so.6:strspn) redirected to 0x48311d0 (_vgnU_ifunc_wrapper)
--9303-- REDIR: 0x4923780 (libc.so.6:stpncpy) redirected to 0x48311d0 (_vgnU_ifunc_wrapper)
--9303-- REDIR: 0x4923720 (libc.so.6:stpcpy) redirected to 0x48311d0 (_vgnU_ifunc_wrapper)
--9303-- REDIR: 0x4924cc0 (libc.so.6:strchrnul) redirected to 0x48311d0 (_vgnU_ifunc_wrapper)
--9303-- REDIR: 0x49238d0 (libc.so.6:strncasecmp_l) redirected to 0x48311d0 (_vgnU_ifunc_wrapper)
--9303-- REDIR: 0x4923320 (libc.so.6:strstr) redirected to 0x48311d0 (_vgnU_ifunc_wrapper)
--9303-- REDIR: 0x4a0b410 (libc.so.6:__strrchr_avx2) redirected to 0x483ea10 (rindex)
--9303-- REDIR: 0x491d110 (libc.so.6:malloc) redirected to 0x483b780 (malloc)
--9303-- REDIR: 0x4a0b220 (libc.so.6:__strchrnul_avx2) redirected to 0x4843540 (strchrnul)
--9303-- REDIR: 0x4a0b5e0 (libc.so.6:__strlen_avx2) redirected to 0x483ef40 (strlen)
--9303-- REDIR: 0x4a0e5f0 (libc.so.6:__memcpy_avx_unaligned_erms) redirected to 0x48429f0 (memmove)
--9303-- REDIR: 0x4a0e5d0 (libc.so.6:__mempcpy_avx_unaligned_erms) redirected to 0x4843660 (mempcpy)
--9303-- REDIR: 0x4a06ae0 (libc.so.6:__strcmp_avx2) redirected to 0x483fed0 (strcmp)
--9303-- REDIR: 0x4a06730 (libc.so.6:__strcspn_sse42) redirected to 0x4843e10 (strcspn)
--9303-- REDIR: 0x4a06870 (libc.so.6:__strpbrk_sse42) redirected to 0x4843da0 (strpbrk)
--9303-- REDIR: 0x4a0aff0 (libc.so.6:__strchr_avx2) redirected to 0x483ebf0 (index)
--9303-- REDIR: 0x4a0cb20 (libc.so.6:__strcpy_avx2) redirected to 0x483f090 (strcpy)
--9303-- REDIR: 0x491d700 (libc.so.6:free) redirected to 0x483c9d0 (free)
--9303-- REDIR: 0x4a0d810 (libc.so.6:__stpcpy_avx2) redirected to 0x4842300 (stpcpy)
--9303-- REDIR: 0x491deb0 (libc.so.6:realloc) redirected to 0x483df30 (realloc)
--9303-- REDIR: 0x4a07440 (libc.so.6:__memchr_avx2) redirected to 0x4840050 (memchr)
==9303==
==9303== HEAP SUMMARY:
==9303==     in use at exit: 1,728 bytes in 53 blocks
==9303==   total heap usage: 301 allocs, 248 frees, 40,956 bytes allocated
==9303==
==9303== Searching for pointers to 53 not-freed blocks
==9303== Checked 81,432 bytes
```

2.1- Ejercicio 1

```
enum { va_eol = -1 };

unsigned int average(int first, ...) {
        unsigned int count = 0;
        unsigned int sum = 0;
        int i = first;
        va_list args;

        va_start(args, first);

        while (i != va_eol) {
                sum += i;
                count++;
                i = va_arg(args, int);
        }
        va_end(args);
        return(count ? (sum / count) : 0);
}
```

1-¿Que hace el siguiente segmento de código?

El código calcula la media del valor de los enteros positivos pasados como argumentos a la función.

2-¿Para que se utiliza la variable va_eol?

va_eol se utiliza para parar el programa, hasta que la función no encuentra el argumento de va_eol (-1) continúa el programa.

2.2- Ejercicio 2

```
#include <stdio.h>


unsigned long long int factorial (unsigned int i){


   if(i <= 1){

      return 1;

   }

   return i * factorial(i - 1);

}


int main (int argc, char* argv[]){


   int i= 12, j=3, f=0;


   if (argc == 1){


      printf("Factorial of %d id %11d\n", i, factorial(i));

   }else{


      j = atoi(argv[1]);

      for (f = 0; f < j; f++){

         printf("Factorial of %d id %11d\n", f, factorial(f));

      }
```

```
    }

    return 0;

}
```

1-¿Que hace el siguiente segmento de código?

El código calcula el factorial de un número introducido por teclado al ejecutar el código.

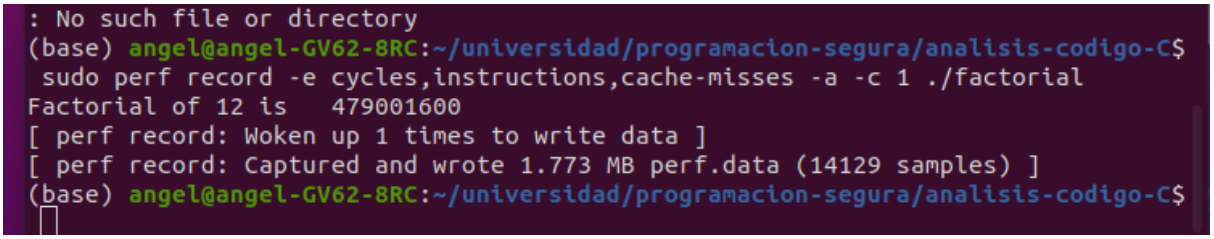
2-Comenta qué reglas/recomendaciones se están rompiendo aquí. También entran reglas pasadas.

La regla que se incumple es la DCL30-C puesto que se está asignando el valor de atoi en la linea 18 y se hace dentro del else, con lo cual al finalizar el else ese valor se va perder, mejor hacerlo en la linea 13 debajo de las inicializaciones.

También se está rompiendo la recomendación DCL02-C. Use visually distinct identifiers, ya que los nombres que se están asignando en la línea 12 de los ints, no son muy identificativos, sería mejor darles un nombre acorde a su función en el código.

Otra recomendación que se está rompiendo es DCL04-C. Do not declare more than one variable per declaration, ya que en la línea 12, los ints se están declarando e inicializando todos en la misma línea, es mejor declarar cada uno en una línea diferente.

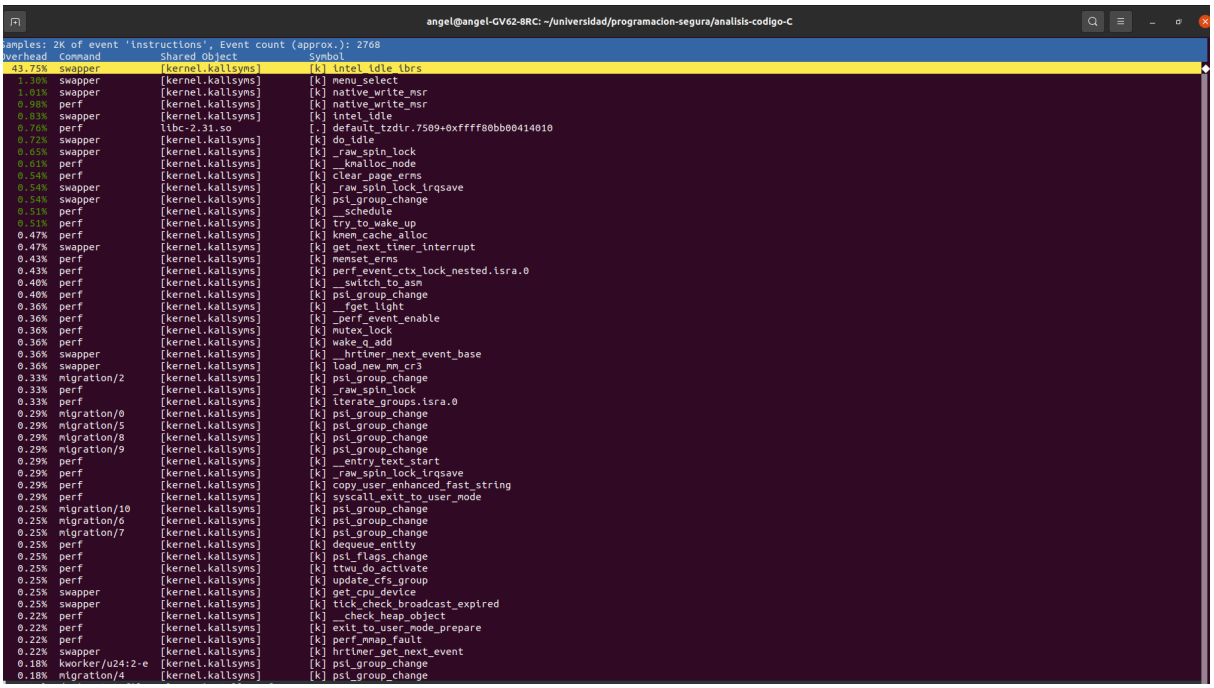
4-El programa permite mostrar el código desensamblado de la aplicación, adjunta alguna captura.



```
Disassembly of section load0:

ffffffffb9971be0 <load0>:
  nop
  push   %rbp
  mov    %rsp,%rbp
  push   %r15
  push   %r14
  push   %r13
  mov    %rsi,%r13
  push   %r12
  mov    %edx,%r12d
  push   %rbx
  nop
→ callq  spec_ctrl_current
  xor    %esi,%esi
  mov    $0x48,%edi
  mov    %rax,%rbx
  mov    %rsi,%rdx
→ callq  native_write_msr
  xchg   %ax,%ax
  xor    %r15d,%r15d
39:  movslq %r12d,%r14
  cmp    $0xa,%r14
↓ ja     121
46:  lea    (%r14,%r14,2),%rax
  lea    (%r14,%rax,4),%rax
  movzbl 0x5b(%r13,%rax,8),%esi
↓ jmpq   103
59: ↓ jmp    70
  nop
  mfence
  mov    %gs:0x1fbc0,%rax
  clflush (%rax)
  mfence
70:  xor    %edx,%edx
  mov    %rdx,%rcx
  mov    %gs:0x1fbc0,%rax
  monitor %rax,%ecx,%edx
  mov    (%rax),%rax
  test   $0x8,%al
↓ jne    9c
  xchg   %ax,%ax
  verw   0x536a2b(%rip)        # 0xffffffffb9ea869c
  mov    $0x1,%ecx
  mov    %rsi,%rax
  mwait  %eax,%ecx
9c:  mov    %gs:0x1fbc0,%rax
  lock   andb   $0xdf,0x2(%rax)
```

5-¿Podrías decir cuál es la instrucción que más tiempo de CPU requiere?



El programa que más CPU está consumiendo es una que se llama intel_idle_ibrs con un consumo del 43.75%.

La siguiente instrucción que más consume es una de swapper con un 1.30% uso del CPU.

2.3- Ejercicio 3

```
(base) angel@angel-GV62-8RC:~/universidad/programacion-segura/analisis-codigo-C$ sudo perf record -e cycles,instructions,cache-misses -a -c 1 ./fib
0
1
1
2
3
5
8
13
21
34
55
89
144
233
377
610
987
1597
2584
4181
6765
10946
17711
28657
46368
75025
121393
196418
317811
514229
832040
1346269
2178309
3524578
5702887
9227465
14930352
24157817
39088169
63245986
102334155
165580141
267914296
433494437
701408733
[ perf record: Woken up 7 times to write data ]
[ perf record: Captured and wrote 4.123 MB perf.data (53858 samples) ]
```

```
 26.03%  swapper          [kernel.kallsyms]       [k] intel_idle_ibrs
  2.09%  fib              fib                     [.] fib
  1.99%  fib              [kernel.kallsyms]       [k] xhci_queue_isoc_tx
  1.68%  fib              [kernel.kallsyms]       [k] handle_tx_event
  1.64%  fib              [kernel.kallsyms]       [k] xhci_irq
  1.38%  fib              [kernel.kallsyms]       [k] memcpy_erms
  1.30%  fib              [kernel.kallsyms]       [k] xhci_get_frame
  1.28%  swapper          [kernel.kallsyms]       [k] menu_select
  1.25%  swapper          [kernel.kallsyms]       [k] native_write_msr
  0.87%  fib              [kernel.kallsyms]       [k] _raw_spin_lock_irqsave
  0.84%  swapper          [kernel.kallsyms]       [k] kmem_cache_free
  0.66%  fib              [kernel.kallsyms]       [k] xhci_ring_ep_doorbell
  0.64%  fib              [kernel.kallsyms]       [k] xhci_update_erst_dequeue
  0.62%  swapper          [kernel.kallsyms]       [k] file_free_rcu
  0.61%  swapper          [kernel.kallsyms]       [k] update_sd_lb_stats.constprop.0
  0.57%  fib              [kernel.kallsyms]       [k] _raw_spin_lock
  0.57%  fib              [kernel.kallsyms]       [k] kfree
  0.55%  kworker/4:2-eve  [kernel.kallsyms]       [k] collect_percpu_times
  0.55%  swapper          [kernel.kallsyms]       [k] rcu_cblist_dequeue
  0.52%  fib              [kernel.kallsyms]       [k] error_entry
  0.51%  fib              [kernel.kallsyms]       [k] prepare_playback_urb
  0.50%  swapper          [kernel.kallsyms]       [k] do_idle
  0.50%  swapper          [kernel.kallsyms]       [k] update_blocked_averages
  0.44%  fib              [kernel.kallsyms]       [k] xhci_td_cleanup
  0.42%  fib              [kernel.kallsyms]       [k] decay_load
  0.42%  swapper          [kernel.kallsyms]       [k] load_balance
  0.41%  fib              [kernel.kallsyms]       [k] __update_load_avg_se
  0.39%  fib              [kernel.kallsyms]       [k] inc_deq
  0.38%  swapper          [kernel.kallsyms]       [k] psi_group_change
  0.37%  fib              [kernel.kallsyms]       [k] irqentry_exit_to_user_mode
```

El programa que más CPU está consumiendo es una que se llama intel_idle_ibrs con un consumo del 26.03%.

La siguientes 6 instrucciones siguientes que más consumen son de fib, con un total de un 10.08%.