

75.03/95.57 Organización del Computador

U4 - CASO DE ESTUDIO INTEL Funciones de C

Manejo de Parámetros

LINUX

rax rdi rsi rdx rcx r8 r9 stack...

type funcName(p1, p2, p3, p4, p5, p6, p7,..., pn)

rax rcx rdx r8 r9 stack...

WINDOWS

Salida por Pantalla

Función **puts**

Imprime un string hasta que encuentra un 0 (cero binario). Agrega el caracter de fin de línea a la salida

```
int  puts(const char *str)
```

LINUX

```
cadena db "Hola",0
```

```
. . .
```

```
mov rdi,cadena
```

```
call puts
```

WINDOWS

```
cadena db "Hola",0
```

```
. . .
```

```
mov rcx,cadena
```

```
call puts
```

Salida por Pantalla

Función **printf**

Convierte a string cada uno de los parámetros y los imprime con el formato indicado por pantalla.

```
int printf(const char *format, arg-list)
```

LINUX

```
msj    db    "Direccion %s %li",0
calle  db    "Paseo Colon",0
num     dq    955

. . .
mov     rdi,msj
mov     rsi,calle
mov     rdx,[num]
call    printf
```

WINDOWS

```
msj    db    "Direccion %s %lli",0
calle  db    "Paseo Colon",0
num     dq    955

. . .
mov     rcx,msj
mov     rdx,calle
mov     r8,[num]
call    printf
```

Salida por Pantalla

Función `printf` - Especificadores de Formato

```
int printf(const char *format, arg-list)
```

			Linux	Windows
%hhi	número entero con signo	base 10	8 bits	-
%hi	número entero con signo	base 10	16 bits	16 bits
%i	número entero con signo	base 10	32 bits	32 bits
%li	número entero con signo	base 10	64 bits	32 bits
%lli	número entero con signo	base 10	-	64 bits
%o	número entero sin signo	base 8	32 bits	32 bits
%x	número entero sin signo	base 16	32 bits	32 bits
%c	caracter			
%s	string			

Ingreso por Teclado

Función **gets**

Lee una serie de caracteres ingresados por teclado hasta que se presiona 'enter' y los almacena en el campo en memoria indicado por parámetro. Agrega un 0 binario al final.

```
char *gets(char *buffer)
```

LINUX

```
texto resb 100  
.  
.  
.  
mov rdi, texto  
call gets
```

WINDOWS

```
texto resb 100  
.  
.  
.  
mov rcx, texto  
call gets
```

Conversión de Formato

Función **sscanf**

Lee una serie de datos desde un string y, de ser posible, los guarda en el formato indicado para cada uno. Retorna la cantidad de datos que se convirtieron correctamente.

```
int  sscanf(const char *buffer, const char *format, arg-list)
```

LINUX

```
numFormat db  "%li",0
. . .
string resb 100
numero resq 1

. . .
mov  rdi, string
mov  rsi, numFormat
mov  rdx, numero
call sscanf
cmp  rax, 1
jl   error
```

WINDOWS

```
numFormat db  "%lli",0
. . .
string resb 100
numero resq 1

. . .
mov  rcx, string
mov  rdx, numFormat
mov  r8, numero
call sscanf
cmp  rax, 1
jl   error
```

Manejo de Archivos - Apertura

Función `fopen`

Abre el archivo especificado en *fileName*, en el modo especificado en *mode*. Retorna un id de archivo o un código de error (valor negativo).

```
FILE * fopen( char * fileName, char * mode )
```

LINUX

```
fileName db  "Miarchivo.txt",0
modo      db  "r+",0
. . .
idArchivo resq 1
. . .
mov  rdi,fileName
mov  rsi,modo
call fopen

cmp  rax,0
jle  errorOPEN
mov  qword[idArchivo],rax
```

WINDOWS

```
fileName db  "Miarchivo.txt",0
modo      db  "r+",0
. . .
idArchivo resq 1
. . .
mov  rcx,fileName
mov  rdx,modo
call fopen

cmp  rax,0
jle  errorOPEN
mov  qword[idArchivo],rax
```


Manejo de Archivos - Apertura

Modos de apertura

```
FILE * fopen( char * fileName, char * mode )
```

Id mode	Tipo Archivo	Operacion	Modo Apertura
r	texto	abre (si existe)	lectura
w	texto	trunca/crea	escritura
a	texto	abre (si existe)/crea	agregar (append)
r+	texto	abre (si existe)	lectura + escritura
w+	texto	trunca/crea	escritura + lectura
a+	texto	abre (si existe)/crea	agregar (append) + lectura
rb	binario	abre (si existe)	lectura
wb	binario	trunca/crea	escritura
ab	binario	abre (si existe)/crea	agregar (append)
rb+	binario	abre (si existe)	lectura + escritura
wb+	binario	trunca/crea	escritura + lectura
ab+	binario	abre (si existe)/crea	agregar (append) + lectura

Manejo de Archivos - Lectura - Texto

Función `fgets`

Lee los siguientes *size* bytes (o hasta encontrar el fin de línea) del archivo identificado por *fp* y los copia en *s*. Retorna la dirección de *s* o un código de error.

```
char *fgets(char *s, int size, FILE *fp)
```

LINUX

```
modo      db  "r",0
. . .
idArchivo  resq  1
registro   resb  81
. . .
mov  rdi,registro
mov  rsi,80
Mov  rdx,[idArchivo]
call fgets

cmp  rax,0
jle  EOF
```

WINDOWS

```
modo      db  "r",0
. . .
idArchivo  resq  1
registro   resb  81
. . .
mov  rcx,registro
mov  rdx,80
Mov  r8,[idArchivo]
call fgets

cmp  rax,0
jle  EOF
```

Manejo de Archivos - Lectura - Binario

Función fread

Lee los siguientes n bloques de tamaño $size$ bytes del archivo identificado por fp y los copia en p . Retorna la cantidad de bloques leídos o un código de error.

```
int fread (void *p, int size, int n, FILE * fp)
```

LINUX

```
modo      db  "rb",0
. . .
idArchivo  resq 1
registro   times 0 resb 22
id         resw 1
nombre     resb 20

. . .
mov  rdi,registro
mov  rsi,5
mov  rdx,1
mov  rcx,[idArchivo]
call fread

cmp  rax,0
jle  EOF
```

WINDOWS

```
modo      db  "rb",0
. . .
idArchivo  resq 1
registro   times 0 resb 22
id         resw 1
nombre     resb 20

. . .
mov  rcx,registro
mov  rdx,5
mov  r8,1
mov  r9,[idArchivo]
call fread

cmp  rax,0
jle  EOF
```

Manejo de Archivos - Escritura - Texto

Función `fputs`

Copia los bytes apuntados por `s` hasta encontrar el 0 binario (este último no se copia) en el archivo identificado por `fp`. Retorna un valor negativo en caso de error o fin de archivo.

```
char *fputs(const char *s, FILE *fp)
```

LINUX

```
modo    db    "w+",0
linea   db    "9557/7503",0
. . .
idArchivo resq 1
. . .
mov     rdi,linea
mov     rsi,[idArchivo]
call    fputs
. . .
```

WINDOWS

```
modo    db    "w+",0
linea   db    "9557/7503",0
. . .
idArchivo resq 1
. . .
mov     rcx,linea
mov     rdx,[idArchivo]
call    fputs
. . .
```

Manejo de Archivos - Escritura - Binario

Función `write`

Copia `n` bloques de `size` bytes en el archivo identificado por `fp`. Retorna la cantidad de bloques escritos o un código de error.

```
int    fwrite(void *p, int size, int n, FILE * fp)
```

LINUX

```
modo      db  "wb+",0
registro   times 0 db ""
          id   dw  2020
          mes  db  "ENE"
          . . .
idArchivo resq  1
          . . .
mov  rdi,registro
mov  rsi,5
mov  rdx,1
mov  rcx,[idArchivo]
call fwrite
          . . .
```

WINDOWS

```
modo      db  "wb+",0
registro   times 0 db ""
          id   dw  2020
          mes  db  "ENE"
          . . .
idArchivo resq  1
          . . .
mov  rcx,registro
mov  rdx,5
mov  r8,1
mov  r9,[idArchivo]
call fwrite
          . . .
```

Manejo de Archivos - Cierre

Función **fclose**

Cierra el archivo identificado por *fp*.

```
void    fclose(FILE *fp)
```

LINUX

```
. . .  
idArchivo  resq    1  
. . .  
mov  rdi,[idArchivo]  
call fclose  
. . .
```

WINDOWS

```
. . .  
idArchivo  resq    1  
. . .  
mov  rcx,[idArchivo]  
call fclose  
. . .
```

Ajustes para llamados a funciones de C

LINUX printf y sscanf

```
. . .  
mov     rdi,msj  
sub     rax,rax  
call    printf  
  
. . .  
mov     rdi,numConvertido  
mov     rdi,format  
mov     rdx,numero  
call    checkAlign  
sub     rsp,[plusRsp]  
call    sscanf  
add     rsp,[plusRsp
```

WINDOWS todas las funciones

```
. . .  
  
sub     rsp,32  
call    <func>  
add     rsp,32  
  
. . .
```

FIN :)