



C Piscine

C 09

*Resumen: Este documento corresponde al enunciado del módulo C 09 de la C Piscine de 42.*

*Versión: 3.3*

# Índice general

I.	Instrucciones	2
II.	Introducción	4
III.	Ejercicio 00 : libft	6
IV.	Ejercicio 01 : Makefile	7
V.	Ejercicio 02 : ft_split	9
VI.	Entrega y evaluación	10

# Capítulo I

## Instrucciones

- Esta página será la única referencia: no te fíes de los rumores.
- ¡Ten cuidado! Los enunciados pueden cambiar en cualquier momento.
- Asegúrate de que tus directorios y archivos tienen los permisos adecuados.
- Debes respetar el procedimiento de entrega para todos tus ejercicios.
- Tus compañeros de piscina se encargarán de corregir tus ejercicios.
- Además de por tus compañeros, también serán corregidos por un programa que se llama la Moulinette.
- La Moulinette es muy estricta a la hora de evaluar. Está completamente automatizada. Es imposible discutir con ella sobre tu nota. Por lo tanto, se extremadamente riguroso para evitar cualquier sorpresa.
- La Moulinette no tiene una mente muy abierta. No intenta comprender el código que no respeta la Norma. La Moulinette utiliza el programa **norminette** para comprobar La Norma en sus archivos. Entiende entonces que es estúpido entregar un código que no pase la **norminette**.
- Los ejercicios han sido ordenados con mucha precisión, del más sencillo al más complejo. En ningún caso se tendrá en cuenta un ejercicio complejo si no se ha conseguido realizar perfectamente un ejercicio más sencillo.
- El uso de una función prohibida se considera una trampa. Cualquier trampa será sancionada con la nota -42.
- Solamente hay que entregar una función `main()` si lo que se pide es un programa.
- La Moulinette compila con los flags `-Wall -Wextra -Werror` y utiliza `cc`.
- Si tu programa no compila, tendrán un 0.
- No puedes dejar en tu directorio ningún archivo que no se haya indicado de forma explícita en los enunciados de los ejercicios.
- ¿Tienes alguna pregunta? Pregunta a tu compañero de la derecha. Si no, prueba con tu compañero de la izquierda.

- Tu manual de referencia se llama `Google / man / Internet / ....`
- ¡No olvides participar en el slack de tu Piscina!
- Lee detenidamente los ejemplos. Podrían exigir cosas que no se especifican necesariamente en los enunciados...
- Razona. ¡Te lo suplico, por Thor, por Odín! Maldita sea.



Para este módulo, la Norminette debe ser ejecutada con el flag `-R CheckForbiddenSourceHeader`. La Moulinette también lo utilizará.

# Capítulo II

## Introducción

Fragmento de Proyecto Hail Mary, de Andy Weir

Toc, toc, toc, toc

-¿Qué?- Me vuelvo a mirarlo.

Señala el símbolo Å que todavía tengo en la mano y luego a mí.

Luego otra vez el Å y otra vez a mí. Lo hace de una forma casi frenética.

-Oh, lo siento- digo. Sostengo el dígito en alto y digo-: Tres.

Hace las manos de jazz. Le devuelvo el gesto.

Eh. Ya que estamos en el tema...

Me quedo quieto un momento, así sabrá que hay un corte en la conversación.

Entonces hago las manos de jazz y digo.

-Sí.

Repito el gesto.

-Sí.

-El vuelve a hacerlo y dice:

-Pp.

Anoto y grabo las frecuencias en mi portátil.

-Está bien. Tenemos “sí” en nuestro vocabulario ahora- digo.

Toc, toc, toc.

Miro. Una vez que sabe que tiene mi atención hace las manos de jazz y dice:

-Pp.

El mismo acorde que antes. -Sí -digo-. Lo tenemos cubierto.

Levanta un dedo un momento. Entonces cierra dos de sus puños y los golpea juntos.

-pp.

-¿Qué?

Ohhh -digo. Soy profesor.

¿Qué le enseñaría a alguien que acaba de aprender la palabra “sí”?

“Eso es “no””.

Al menos eso espero.

Cierro los puños y los golpeo juntos.

-No.

-Pp dice. Verifico el portátil. Acaba de decir sí.

Espera. ¿Significa eso que no es no? ¿Es eso otro sí? Ahora estoy confundido.

¿No? -pregunto.

-No- dice en eridiano.

Entonces «sí»? -No, sí.

¿Sí? -No. No. -¿Sí, sí?


¡No! -Cierra un puño hacia mí, claramente frustrado.

Basta de esta rutina de Abbott y Costello interespecies.

Por desgracia, este enunciado no tiene nada que ver con el aprendizaje del eridiano.

# Capítulo III

## Ejercicio 00 : libft

	Ejercicio: 00
libft	
Directorio de entrega: <i>ex00/</i>	
Archivos a entregar: <code>libft_creator.sh</code> , <code>ft_putchar.c</code> , <code>ft_swap.c</code> , <code>ft_putstr.c</code> , <code>ft_strlen.c</code> , <code>ft_strcmp.c</code>	
Funciones autorizadas: <code>write</code>	

- Crea tu biblioteca `ft`. Se llamará `libft.a`.
- Un script shell denominado `libft_creator.sh` compilará correctamente los archivos fuente y creará tu biblioteca.
- Esta biblioteca debe incluir todas las funciones que se indican a continuación:


```
void    ft_putchar(char c);
void    ft_swap(int *a, int *b);
void    ft_putstr(char *str);
int     ft_strlen(char *str);
int     ft_strcmp(char *s1, char *s2);
```

- Ejecutaremos el comando siguiente:

```
sh libft_creator.sh
```

# Capítulo IV

## Ejercicio 01 : Makefile

	Ejercicio: 01
Makefile	
Directorio de entrega: <i>ex01/</i>	
Archivos a entregar: <b>Makefile</b>	
Funciones autorizadas: <b>Ninguna</b>	

- Crea el **Makefile** que compile una librería **libft.a**.
- Tu **makefile** debe mostrar cada comando que ejecute.
- Tu **makefile** no debe ejecutar ningún comando inútil.
- El **Makefile** irá a buscar los archivos fuente al directorio **srcs**.
- Estos archivos fuente serán: **ft\_putchar.c**, **ft\_swap.c**, **ft\_putstr.c**, **ft\_strlen.c**, **ft\_strcmp.c**
- El **Makefile** irá a buscar los archivos headers al directorio **includes**.
- Estos archivos headers serán: **ft.h**
- Tendrás que compilar los archivos C utilizando **cc** y las opciones flag **-Wall -Wextra -Werror**, en ese orden.
- La librería estará en la raíz del ejercicio.
- Los archivos **.o** tendrán que estar al lado de tus archivos **.c** correspondientes.
- El **Makefile** también tendrá que implementar reglas **clean**, **fclean**, **re**, la regla **all** y, por supuesto, **libft.a**.
- Ejecutar solo **make** debe ser equivalente a **make all**.
- La regla **all** tendrá que hacer lo mismo que **make libft.a**.
- La regla **clean** tendrá que retirar todos los archivos temporales que se hayan generado.




- La regla `fclean` hará lo mismo que un `make clean` y borrará también el binario creado al ejecutar `make all`.
- La regla `re` hará lo mismo que un `make fclean` seguido de un `make all`.
- Tu makefile no debe compilar los archivos inútilmente.
- Solo recogeremos tu Makefile y lo probaremos con nuestros archivos.



¡Ten cuidado con los wildcards!

# Capítulo V

## Ejercicio 02 : ft\_split

	Ejercicio: 02
	ft_split
	Directorio de entrega: <i>ex02/</i>
	Archivos a entregar: <b>ft_split.c</b>
	Funciones autorizadas: <b>malloc</b>

- Escribe una función que divida una cadena de caracteres en función de otra cadena de caracteres.
- Habrá que utilizar cada carácter de la cadena **charset** como separador.
- La función devuelve una tabla donde cada uno de tus elementos contiene la dirección de una cadena de caracteres comprendida entre dos separadores. El último elemento de la tabla tendrá que ser igual a 0 para indicar el final de la tabla.
- Tu tabla no puede tener cadenas vacías. Saca las conclusiones pertinentes.
- La cadena que se transmitirá no será modificable.
- El prototipo de la función deberá ser el siguiente:

```
char **ft_split(char *str, char *charset);
```

# Capítulo VI

## Entrega y evaluación

Entrega tu proyecto en tu repositorio `Git` como de costumbre. Solo el trabajo entregado en el repositorio será evaluado durante la defensa. No dudes en comprobar varias veces los nombres de los archivos para verificar que sean correctos.



Sólo necesitas entregar los archivos requeridos por el enunciado de este proyecto.