

# Sistemas Operativos

sesión 2: problemas lenguaje C

Grado en Ingeniería Informática

Universidad Carlos III de Madrid

# Agenda



Linux



Punteros y  
parámetros



Ejercicios



# Agenda



Linux



Punteros y  
parámetros



Ejercicios



# Contenidos



- Entorno local y remoto
- Sesión de trabajo básica
- Navegación por el sistema de ficheros

# Contenidos



- **Entorno local y remoto**
- Sesión de trabajo básica
- Navegación por el sistema de ficheros

# Ordenadores de trabajo: local

- Uso de los ordenadores del Laboratorio del Departamento de Informática:
  - 4.0.F16, 4.0.F18 + 1.0.A01, **1.0.A02**, 1.0.H02
- Solicitud de apertura de cuenta en:  
[http://www.lab.inf.uc3m.es/servicios/apertura\\_cuentas](http://www.lab.inf.uc3m.es/servicios/apertura_cuentas)
- Con el usuario y clave mandados como respuesta a la solicitud, se puede iniciar una sesión de trabajo.

# Ordenadores de trabajo: remoto

- Desde otro ordenador es posible acceder a los ordenadores con una conexión remota:
  - `guernika.lab.inf.uc3m.es`
- Necesario un cliente SSH:



PuTTY



SSH  
Secure  
Shell



Cyberduck



Fugu

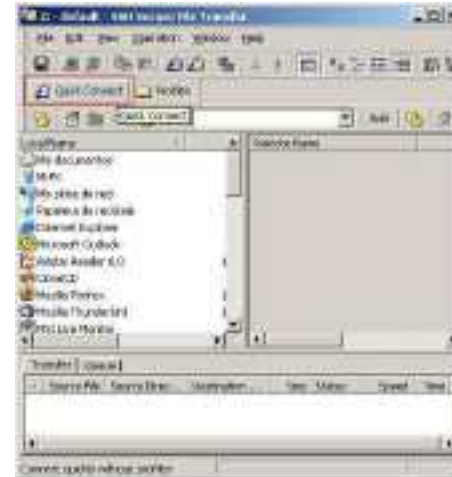
...

# Copia de ficheros entre ordenador local y remoto

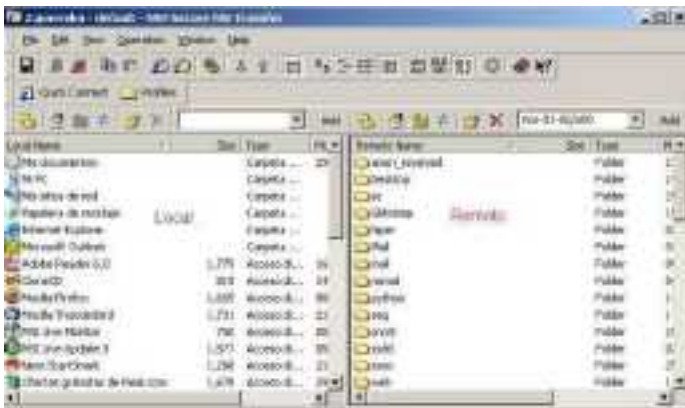
1



2



4




3





# Problemas con la cuenta de los laboratorios del Dpto. de Informática

- Revisar las preguntas frecuentes y tutoriales:
  - <http://www.lab.inf.uc3m.es/informacion/faq>
  - <http://www.lab.inf.uc3m.es/informacion/tutoriales>
- Enviar un correo a la dirección de soporte:
  - lab  lab.inf.uc3m.es

# Contenidos



- Entorno local y remoto
- **Sesión de trabajo básica**
- Navegación por el sistema de ficheros

# Sesión de trabajo

- `ssh guernika.lab.inf.uc3m.es -X -l a00xxxxx`
  - Nos pedirá el usuario y la clave, si todo está correcto iniciará una sesión de trabajo.
- `passwd`
  - Cambiar la clave, para lo que pedirá la antigua clave y dos veces la nueva clave.
- `exit`
  - Finalizar la sesión de trabajo.

# Información de recursos

- `quota -vs`
  - Nos indicará el espacio de disco usado (en capacidad y número de ficheros) y el máximo que nos está permitido usar de la cuenta.
- `du -mh`
  - Indicará el espacio usado del directorio actual (incluyendo subdirectorios)

# Obtener ayuda

- `man <mandato>`
  - Muestra la ayuda del mandato.
  - Con barra espaciadora se avanza y con 'b' se retrocede; para salir hay que usar la letra 'q'

# Contenidos



- Entorno local y remoto
- Sesión de trabajo básica
- **Navegación por el sistema de ficheros**

# Navegación por directorios

- `ls -las`
  - Muestra los archivos y subdirectorios del directorio actual de trabajo.
- `pwd`
  - Imprime el directorio actual de trabajo.
- `cd <directorio>`
  - Cambia el directorio actual de trabajo al indicado por parámetro
  - Ej.: `cd /tmp`, `cd ..`
- `cd`
  - Vuelve al directorio inicial de la cuenta de trabajo.

# Analizar ficheros

- `file <fichero>`
  - Indica el tipo de fichero (texto, binario, etc.)
- `cat <fichero>`
  - Muestra el contenido del fichero en pantalla.
- `more <fichero>`
  - Muestra el contenido del fichero pantalla a pantalla.
  - Con barra espaciadora se avanza y con 'b' se retrocede; para salir hay que usar la letra 'q'



# Modificando el sistema de ficheros



- `mkdir <directorio>`
  - Crea un directorio con el nombre indicado.
- `rmdir <directorio>`
  - Borra un directorio.

# Modificando el sistema de ficheros



- `cp <fichero origen> <fichero destino>`
  - Copia un fichero.
- `mv <fichero origen> <fichero destino>`
  - Mueve un fichero de directorio y/o cambia el nombre.
- `rm <fichero>`
  - Borra un fichero.
  - **ATENCIÓN:** no es posible desborrar ficheros en Linux.

# Agenda



Linux



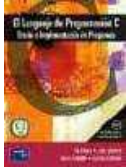
Punteros y  
parámetros



Ejercicios

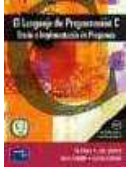


# Contenidos



- Punteros
- Paso de parámetros

# Contenidos

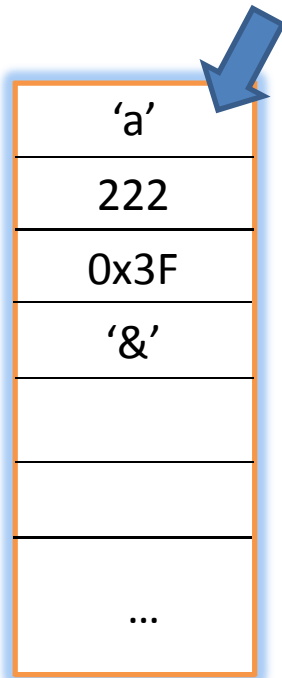


- Punteros
- Paso de parámetros

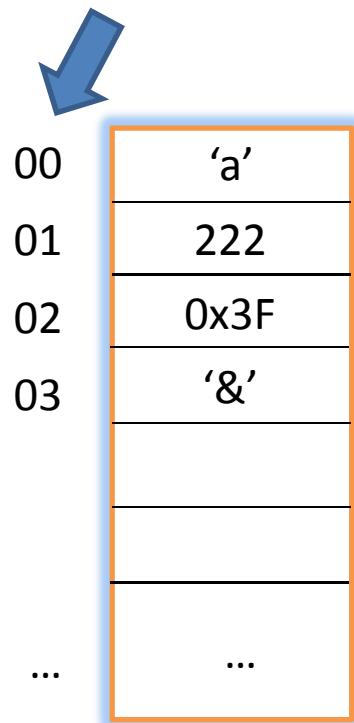
# Dirección y contenido

- Contenido

- Elemento guardado en una posición de memoria



# Dirección y contenido



- Contenido

- Elemento guardado en una posición de memoria

- Dirección

- Identificador de la posición de memoria
- **Puntero** representa la dirección de un elemento de un tipo

# Dirección y contenido: ejemplo

```
#include <stdio.h>

int main ( int argc,
           char *argv[] )
{
    int i = 3;
    int *pi ;

    i = 3;
    pi = &i ;
    printf( "%d\n", *pi) ;

    return 0;
}
```



int \*

– Puntero a entero



& ...

– Dirección de



\* ...

– Valor contenido en



# Dirección y contenido: ejemplo

```
#include <stdio.h>

int main ( int argc,
           char *argv[] )
{
    int i = 3;
    int *pi ;

    i = 3;
    pi = &i ;
    printf( "%d\n", *pi) ;

    return 0;
}
```



**gcc -Wall -g -o e11 e11.c**

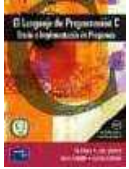
- **Wall:**  
mostrar advertencias
- **g:**  
añadir información de depuración
- **o:**  
establecer el nombre del ejecutable



**./e11**

- El directorio actual (.) no está en la variable PATH

# Contenidos



- Punteros
- Paso de parámetros

# Paso de parámetros

```
void prueba2 ( int j, char c, float f, int pj )  
{  
    /* ... */  
}
```

# Paso de parámetros

```
i = 10 ;  
float PI = 3.14 ;  
  
prueba2 ( i,      'a',      PI,      &i ) ;  
...
```

```
void prueba2 ( int j, char c, float f, int pj )  
{  
    /* ... */  
}
```

# Paso de parámetros

```
i = 10 ;  
float PI = 3.14 ;
```

```
prueba2 ( i,      'a',      PI,      &i ) ;  
...
```

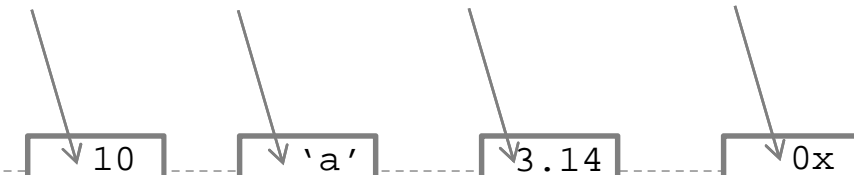
```
void prueba2 ( , , ,  )  
{  
    /* ... */  
}
```

1) Se crea en pila las variables formales

# Paso de parámetros

```
i = 10 ;  
float PI = 3.14 ;
```


```
prueba2 ( i,      'a',      PI,      &i ) ;  
...
```



```
void prueba2 ( int j, char c, float f, int pj )  
{  
    /* ... */  
}
```

2) Se copia el valor de los parámetros reales

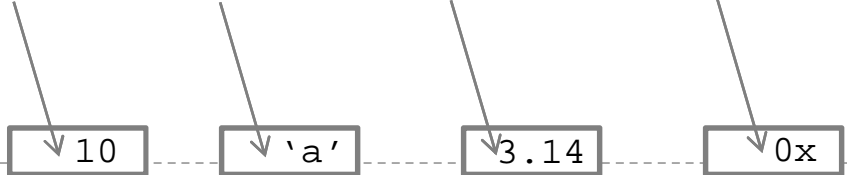
# Paso de parámetros



Siempre se realiza una copia de los parámetros

```
i = 10 ;  
float PI = 3.14 ;
```

```
prueba2 ( i,      'a',      PI,      &i ) ;  
...
```



```
void prueba2 ( int j, char c, float f, int pj )  
{  
    /* ... */  
}
```

# Paso de parámetros

Paso de parámetro  
por **valor**

```
#include <stdio.h>

int main (void)
{
    int i = 10;

    /* ... */
    inc(i) ;
    /* ... */
}
```

```
void inc ( int j )
{
    j = j + 1 ;
}
```



# Paso de parámetros

Paso de parámetro  
por **valor**

```
#include <stdio.h>
```

```
int main (void)
```

```
{
```

```
    int i = 10;
```

```
    /* ... */
```

```
    inc(i) ;
```

```
    /* ... */
```

```
}
```

1) se copia  
i en j

10

```
void inc ( int j )
```

```
{
```

```
    j = j + 1 ;
```

```
}
```

# Paso de parámetros

Paso de parámetro  
por **valor**

```
#include <stdio.h>
```

```
int main (void)
```

```
{
```

```
    int i = 10;
```

```
    /* ... */
```

```
    inc(i) ;
```

```
    /* ... */
```

```
}
```

1) se copia  
i en j

10

```
void inc ( int j )
```

```
{
```

```
    j = j + 1 ;
```

```
}
```

2) se modifica  
j (la copia)

# Paso de parámetros

Paso de parámetro  
por **valor**

Paso de parámetro  
por **referencia**

```
#include <stdio.h>
```

```
int main (void)  
{
```

```
    int i = 10;
```

```
    /* ... */
```

```
    inc(i) ;
```

```
    /* ... */
```

```
}
```

1) se copia  
i en j

10

```
void inc ( int j )
```

```
{
```

```
    j = j + 1 ;
```

```
}
```

2) se modifica  
j (la copia)



# Paso de parámetros

Paso de parámetro  
por **valor**

```
#include <stdio.h>
```

```
int main (void)  
{
```

```
    int i = 10;
```

```
    /* ... */
```

```
    inc(i) ;
```

```
    /* ... */
```

```
}
```

1) se copia  
i en j

10

```
void inc ( int j )
```

```
{
```

```
    j = j + 1 ;
```

```
}
```

2) se modifica  
j (la copia)

Paso de parámetro  
por **referencia**

```
#include <stdio.h>
```

```
int main (void)  
{
```

```
    int i = 3;
```

```
    /* ... */
```

```
    inc(&i) ;
```

```
    /* ... */
```

```
}
```

```
void inc ( int *j )
```

```
{
```

```
    *j = *j + 1 ;
```

```
}
```

# Paso de parámetros

Paso de parámetro  
por **valor**

```
#include <stdio.h>
```

```
int main (void)  
{
```

```
    int i = 10;
```

```
    /* ... */
```

```
    inc(i) ;
```

```
    /* ... */
```

```
}
```

1) se copia  
i en j

10

```
void inc ( int j )
```

```
{
```

```
    j = j + 1 ;
```

```
}
```

2) se modifica  
j (la copia)

Paso de parámetro  
por **referencia**

```
#include <stdio.h>
```

```
int main (void)  
{
```

```
    int i = 3;
```

```
    /* ... */
```

```
    inc(&i) ;
```

```
    /* ... */
```

```
}
```

1) se copia  
&i en j

&i

```
void inc ( int *j )
```

```
{
```

```
    *j = *j + 1 ;
```

```
}
```

# Paso de parámetros

Paso de parámetro  
por **valor**

```
#include <stdio.h>
```

```
int main (void)  
{
```

```
    int i = 10;
```

```
    /* ... */
```

```
    inc(i) ;
```

```
    /* ... */
```

```
}
```

1) se copia  
i en j

10

```
void inc ( int j )
```

```
{
```

```
    j = j + 1 ;
```

```
}
```

2) se modifica  
j (la copia)

Paso de parámetro  
por **referencia**

```
#include <stdio.h>
```

```
int main (void)  
{
```

```
    int i = 3;
```

```
    /* ... */
```

```
    inc(&i) ;
```

```
    /* ... */
```

```
}
```

1) se copia  
&i en j

&i

```
void inc ( int *j )
```

```
{
```

```
    *j = *j + 1 ;
```

```
}
```

2) se modifica  
i a través de \*j

# Paso de parámetros: ejemplo

```
#include <stdio.h>

void inc ( int *j )
{
    *j = *j + 1 ;
}

int main (void)
{
    int i = 3;

    inc(&i) ;
    printf("%d\n",i) ;

    return 0;
}
```

- La función *inc* incrementa el valor pasado por referencia en *j*
- La función *main* define una variable *i*, incrementa su valor y lo imprime

# Paso de parámetros: ejemplo

```
#include <stdio.h>

void inc ( int *j )
{
    *j = *j + 1 ;
}

int main (void)
{
    int i = 3;

    inc(&i) ;
    printf("%d\n",i) ;

    return 0;
}
```



**gcc -Wall -g -o e12 e12.c**

- **Wall:**  
mostrar advertencias
- **g:**  
añadir información de depuración
- **o:**  
establecer el nombre del ejecutable



**./e12**

- El directorio actual (.) no está en la variable PATH



# Paso de parámetros: ejemplo

```
#include <stdio.h>

void inc ( int *j )
{
    *j = *j + 1 ;
}

int main (void)
{
    int i = 3;
    inc( i) ;
    printf( "%d\n", i) ;

    return 0;
}
```



**gcc -Wall -g -o e13 e13.c**


- **Wall:**  
mostrar advertencias
- **g:**  
añadir información de depuración
- **o:**  
establecer el nombre del ejecutable



**./e13**

- El directorio actual (.) no está en la variable PATH

# Paso de parámetros: ejemplo



```
acaldero@phoenix:/tmp$ ./e13
Violación de segmento
acaldero@phoenix:/tmp$ gdb e13
GNU gdb (Ubuntu/Linaro 7.2-1ubuntu1) 7.2
Copyright (C) 2010 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 "i686-linux-gnu".
Para las instrucciones de informe de errores, vea:
<http://www.gnu.org/software/gdb/bugs/>...
Leyendo símbolos desde /tmp/e13...hecho.
(gdb) run
Starting program: /tmp/e13

Program received signal SIGSEGV, Segmentation fault.
0x080483ca in inc (j=0x3) at e13.c:5
5          *j = *j + 1 ;
(gdb) █
```

# Paso de parámetros: ejemplo

```
#include <stdio.h>

void inc ( int *j )
{
    if (j==NULL)
        printf("j=NULL\n") ;
    else *j = *j + 1 ;
}

int main (void)
{
    int i = 3;

    inc(i) ;
    printf("%d\n",i) ;

    return 0;
}
```



**gcc -Wall -g -o e14 e14.c**

- **Wall:**  
mostrar advertencias
- **g:**  
añadir información de depuración
- **o:**  
establecer el nombre del ejecutable



**./e14**

- El directorio actual (.) no está en la variable PATH

# Agenda



Linux



Punteros y  
parámetros



Ejercicios



# Contenidos



- Problemas de lenguaje C:
  - Sumar y dividir enteros
  - Calculadora de fracciones

# Contenidos



- Problemas de lenguaje C:
  - **Sumar y dividir enteros**
  - Calculadora de fracciones

# Sumar y dividir enteros

- Realizar un programa en C que:
  - Implemente las funciones:
    - `void sumar ( int resultado, int a, int b ) ;`
    - `void dividir ( int * resultado, int a, int b ) ;`
  - Implementar la función principal *main* de manera que:
    - Realice la suma de 100 y 350 e imprima el resultado.
    - Realice la división entera de 450 entre 40 e imprima igualmente el resultado.

# Contenidos



- Problemas de lenguaje C:
  - Sumar y dividir enteros
  - **Calculadora de fracciones**



# Calculadora de fracciones

- Realizar un programa en C que:
  - Implemente las funciones:
    - void sumar ( struct fraq \*resultado,  
struct fraq a, struct fraq b ) ;
    - void restar ( struct fraq \*resultado,  
struct fraq a, struct fraq b ) ;
    - void multiplicar ( struct fraq \*resultado,  
struct fraq a, struct fraq b ) ;
    - void dividir ( struct fraq \*resultado,  
struct fraq a, struct fraq b ) ;
  - Implementar la función principal *main* que:
    - Use las funciones anteriores y pruebe que funcionan perfectamente en todos los casos.

# Sistemas Operativos

sesión 2: problemas lenguaje C

Grado en Ingeniería Informática

Universidad Carlos III de Madrid