Programación Avanzada (TC2025)

Tema 1. Programación en lenguaje C

Instituto Tecnológico y de Estudios Superiores de Monterrey. Campus Santa Fe
Departamento de Tecnologías de Información y Electrónica
Dr. Vicente Cubells (vcubells@itesm.mx)

Temario

- Introducción a la materia
- Introducción al lenguaje C
- Evolución del lenguaje
- Diferencias con Java y C#
- Tipos de datos
- Operadores
- Algunas estructuras de control
- Introducción a las funciones
- Sintaxis
- Edición, compilación y enlace
- Estructura de un programa en C
- Ejemplos

Introducción a TC2025

Tema 1. Programación en lenguaje C

Tema 2. Arquitectura de un sistema operativo

Tema 3. Administración de procesos

Tema 4. Eventos y señales

Tema 5. Programación concurrente

Tema 6. Programación paralela

Sistema de evaluación…

Fechas de exámenes

Examen	Fecha
Primer parcial	Jueves 17 de Septiembre de 2015
Segundo parcial	Jueves 29 de Octubre de 2015
Examen final	**Viernes 4 de Diciembre de 2015 a las 12:00 horas**

Sistema de evaluación

Ponderaciones

Evaluación	Porcentaje	Actividad	Porcentaje
Primer Parcial	20%	Tarea 1	15%
		Tarea 2	15%
		Examen Parcial 1	70%
Segundo Parcial	20%	Tarea 3	15%
		Tarea 4	15%
		Examen Parcial 2	70%
Final	55%	Tarea 5	10%
		Tarea 6	10%
		Proyecto final	40%
		Examen Final	**40%**
Semana-i	5%		
Total	100%		

Requerimientos para las tareas, ejercicios de clases y el proyecto

- Contar con una Raspberry Pi o una Beaglebone Black

Proyecto final...

- Se desarrollará en equipos de dos (2) personas.
- La codificación se realizará en C, utilizando OpenMP y MPI.
- El proyecto debe funcionar en un clúster híbrido de cualquier tamaño compuesto por Raspberry Pi + laptops (al menos 2 RPi + 2 laptops). Mientras más componentes diferentes tenga el clúster, mayor será la complejidad del proyecto y esto formará parte de su calificación.
- La entrega final incluye una presentación del proyecto en la última semana.
- Se debe subir el código fuente, la documentación y todo lo relacionado con el proyecto a GitHub.

Proyecto final...

Concepto	%
Originalidad de la solución	5%
Complejidad del software	20%
Funcionalidad y operatividad del software	50%
Calidad del software	10%
Presentación y exposición (incluye presentación en **Exposición de Proyectos de Ingeniería** y su **subida a GitHub**)	15%
Proyecto final	**100%**

Introducción…

- C define el comienzo de la era moderna en la programación
- C fue creado en 1970´s
- Contrario a otros lenguajes, fue diseñado, implementado y desarrollado por programadores reales
- C es un lenguaje estructurado
- Problemas de compatibilidad llevaron a una estandarización en 1989
  - Existe una revisión de 1999 (C99) y otra del **2011** (C11)

Cuando programe en C, utilizará:

- Funciones de la biblioteca estándar de C
- Funciones creadas por usted mismo
- Funciones creadas por otras personas y disponibles para usted

Type	Minimal Range
char	–127 to 127
unsigned char	0 to 255
signed char	–127 to 127
int	–32,767 to 32,767
unsigned int	0 to 65,535
signed int	Same as **int**
short int	–32,767 to 32,767
unsigned short int	0 to 65,535
signed short int	Same as **short int**
long int	–2,147,483,647 to 2,147,483,647
signed long int	Same as **long int**
unsigned long int	0 to 4,294,967,295
float	1E–37 to 1E+37, with six digits of precision
double	1E–37 to 1E+37, with ten digits of precision
long double	1E–37 to 1E+37, with ten digits of precision

Operadores

- Aritméticos: +, -, *, /, %
- Operador ternario: ? :
- Comparación: <, >, <=, >=, ==, !=
- Lógicos: &&, ll, !
- Incremento y decremento: ++, --
- Asignación: =
- Desplazamiento de bits: <<, >>

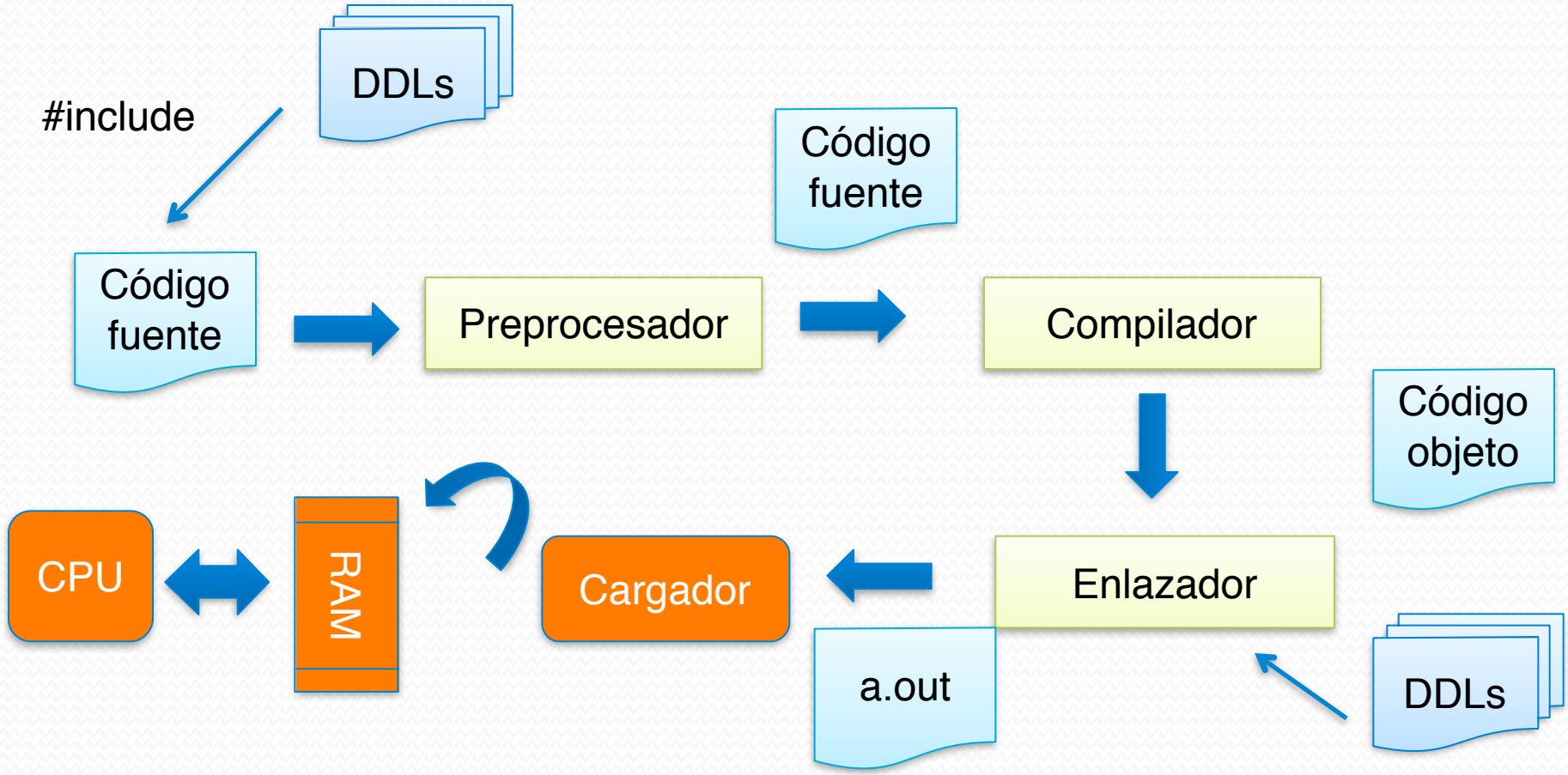
Introduciendo algunos elementos

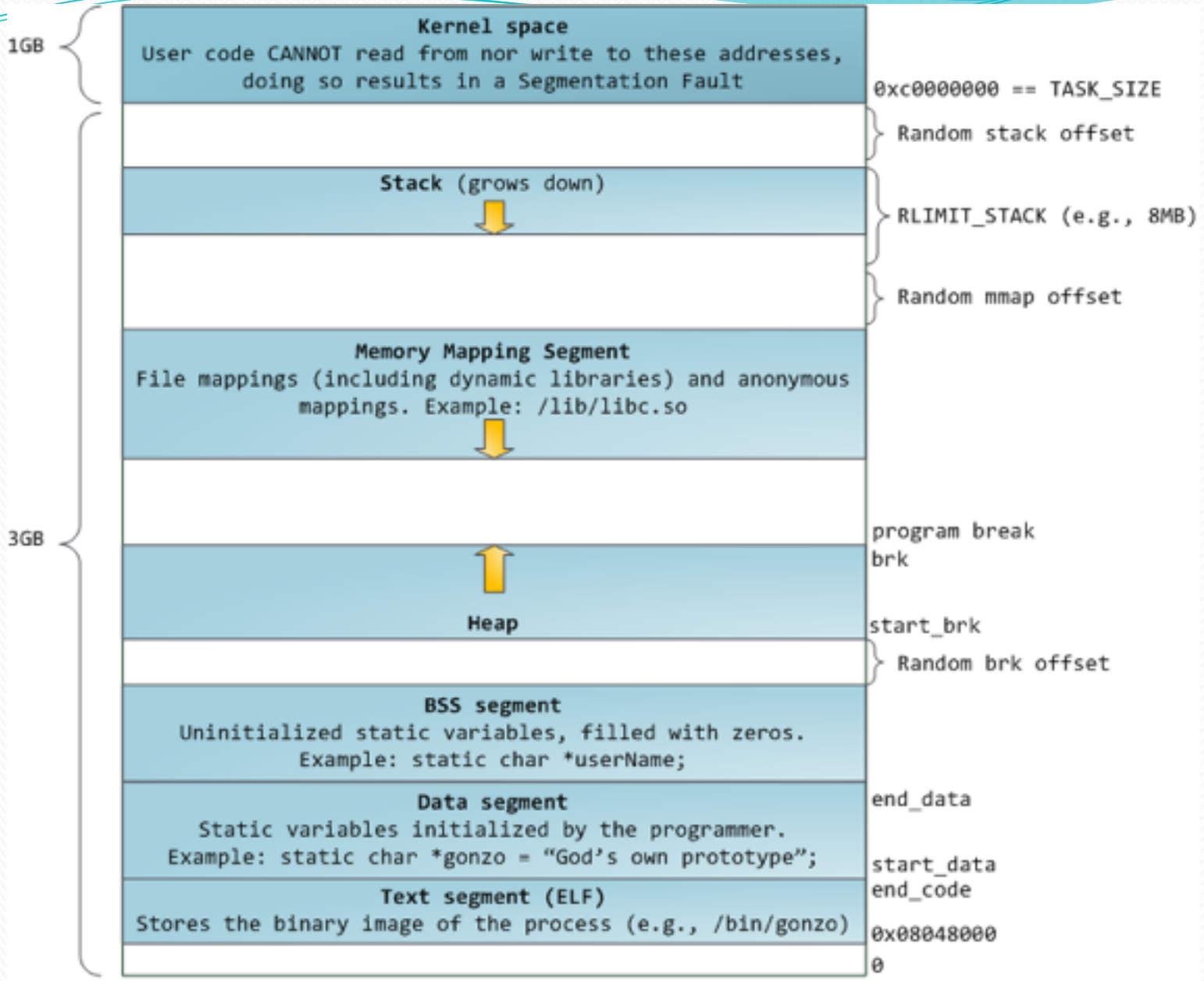
- Estructuras de control
  - *if, switch*
  - *for, while, do..while*
  - *break, continue*
- Uso de float o double en expresiones monetarias
  - 14.234 + 18.673
  - 14.23 + 18.67 = 32.90 (Se imprime 32.91)
- Funciones
  - Func()

Ejemplo de un programa en C

```c
#include <stdio.h>

/* la función main inicia la ejecución del programa */
int main()
{
   int entero1; /* primer número introducido por el usuario  */
   int entero2; /* segundo número introducido por el usuario */
   int suma;    /* variable en la cual se almacena la suma   */

   printf( "Introduzca el primer entero\n" ); /* indicador */
   scanf( "%d", &entero1 );           /* lee un entero */

   printf( "Introduzca el segundo entero\n" ); /* indicador */
   scanf( "%d", &entero2 );           /* lee un entero */

   suma = entero1 + entero2; /* asigna el total a suma */

   printf( "La suma es %d\n", suma ); /* imprime la suma */

   return 0; /* indica que el programa terminó con éxito */

} /* fin de la función main */
```

Edición, compilación y enlace




1GB
3GB
Kernel space
User code CANNOT read from nor write to these addresses, doing so results in a Segmentation Fault
0xc0000000 == TASK_SIZE
Random stack offset
Stack (grows down)
RLIMIT_STACK (e.g., 8MB)
Random mmap offset
Memory Mapping Segment
File mappings (including dynamic libraries) and anonymous mappings. Example: /lib/libc.so
program break
brk
Heap
start_brk
Random brk offset
BSS segment
Uninitialized static variables, filled with zeros.
Example: static char *userName;
Data segment
Static variables initialized by the programmer.
Example: static char *gonzo = "God's own prototype";
end_data
start_data
Text segment (ELF)
Stores the binary image of the process (e.g., /bin/gonzo)
end_code
0x08048000
0

Tres tipos de almacenamiento…

- Almacenamiento automático (stack)
  - Ejemplos
    - Variables locales
    - Argumentos de las funciones
  - Se crean automáticamente al inicio del bloque
  - Se destruyen automáticamente al finalizar el bloque
  - Su valor predeterminado es indeterminado

Tres tipos de almacenamiento…

- Almacenamiento estático
  - Ejemplos
    - Variables estáticas
  - La dirección de un objeto estático es la misma durante la ejecución del programa
  - Son inicializados a ceros binarios

Tres tipos de almacenamiento…

- Ejemplo de Almacenamiento estático

```
int num;

int func() {
        static int calls;          // Se incializa
en 0
        return calls++;
}
```

Tres tipos de almacenamiento

- Almacenamiento dinámico (heap)
  - Ejemplos:
    - variables creadas con la función `malloc`
  - Persisten en memoria hasta que sean liberadas explícitamente mediante la función `free`
    - La memoria ocupada por este tipo de variables no es liberada al SO automáticamente al terminar la ejecución del programa
  - Su dirección de memoria se determina en tiempo de ejecución

Resumiendo

- C es la base de otros lenguajes de programación
- La sintaxis es parecida a Java, PHP y otros lenguajes que se derivaron de C
- Es el lenguaje de más bajo nivel por encima de ensamblador…

Tipos de datos compuestos

- Estructuras
- Uniones
- Enumeraciones

Estructuras…

- Agrupación de datos de tipos diferentes
- Registros que agrupan datos primitivos para modelar un objeto complejo en forma de registro

```
struct libro
{
        int paginas;
        char titulo[25];
        char autor[30];
} programacion;
```

```
struct libro
{
        int paginas;
        char titulo[25];
        char autor[30];
};

struct libro programación;
```

Estructuras…

- Agrupación de datos de tipos diferentes
- Registros que agrupan datos primitivos para modelar un objeto complejo en forma de registro

```
typedef struct
{
        int paginas;
        char titulo[25];
        char autor[30];
} libro;

libro programacion;
```

Estructuras…

- Ejemplo de acceso a los miembros

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

typedef struct
{
        int paginas;
        char titulo[25];
        char autor[30];
} libro;

libro programacion;

int main()
{
        programacion.paginas = 350;
        strcpy(programacion.titulo, “Programación Avanzada”);

        return 0;
}
```

Estructuras…

- Estructuras anidadas

```
struct autor
{
        int edad;
        char nombre[25];
        char apellidos[30];
};
```

```
struct libro
{
        int paginas;
        char titulo[25];
        struct autor escritor;
};

struct libro programación;
```

Estructuras

- Estructuras anidadas

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

/* Aquí van las declaraciones de las estructuras anteriores */

int main()
{
        programacion.paginas = 350;
        strcpy(programacion.titulo, “Programación Avanzada”);
        strcpy(programacion.escritor.nombre, “Deitel”);

        printf(“El autor es: %s \n”, programacion.escritor.nombre);

        return 0;
}
```

Uniones…

- Similar a las estructuras
- En una estructura cada miembro ocupa un área de memoria diferente
- En una unión todos los miembros ocupan el mismo espacio de memoria
  - Modificar uno, significa modificar el otro
  - Sólo se puede acceder a un miembro a la vez
  - Ocupa el espacio del miembro de mayor tamaño

Uniones

- Ejemplo

```
#include <stdio.h>

union libro
{
        int paginas;
        char titulo[25];
        char autor[30];
} programacion;

int main()
{
        programacion.paginas = 350;
        strcpy(programacion.titulo, “Programación Avanzada”);
        strcpy(programacion.autor, “Deitel”);

        printf(“El título es: %s \n”, programacion.titulo);

        return 0;
}
```

Ejemplo

- Programar una aplicación que muestre la diferencia entre estructuras y uniones

Enumeraciones…

- Tipo de dato definido con constantes enteras

```
enum Nombre
{
        enumerador1 = valor_constante1,
        enumerador2 = valor_constante2,
        ...
        enumeradorn = valor_constanten,
};
```

Enumeraciones…

- Ejemplos

```
enum Boolean
{
        FALSE,
        TRUE
};

int main()
{
        enum Boolean existe = FALSE;
        while (existe != TRUE )
        {
                /* Hacer algo */
        }
        return 0;
}
```

Enumeraciones

- Ejemplos

```
enum DiasSemanas
{
        Domingo = 2,
        Lunes,
        Marte,
        Miercoles,
        Jueves,
        Viernes,
        Sabado
};

int main(int argc, char** argv)
{
        enum DiasSemanas dia;

        for (dia = Domingo; dia <= Sabado; dia++)
        {
                printf("%d ", dia);
        }

        return 0;
}
```

Resumiendo

- Las estructuras, uniones y enumeraciones son tipos de datos que nos permiten agrupar tipos primitivos en forma de registros
- En las estructuras cada miembro tiene su propio espacio de memoria
- En las uniones todos los miembros comparten el mismo espacio de memoria
- Las enumeraciones son agrupaciones de constantes enteras, que permiten asociar un valor numérico a un identificador de texto

Ejercicio

- Realice un programa que permita entrar un listado de personas (de cada persona se conoce su nombre, apellidos y edad) y permita:
  - Determinar la edad promedio
  - La persona más joven
  - La persona más vieja
  - Todas las personas que se encuentran en un rango de edades