Introducción al lenguaje C

Juan C. Dueñas <jcduenas@dit.upm.es>

Joaquín Seoane <jseoane@dit.upm.es>

T. de Miguel <tmiguel@dit.upm.es>

Dpto. Ingeniería de Sistemas Telemáticos

Universidad Politécnica de Madrid

Objetivos

Leer y escribir programas en ANSI C

1. Tipos de datos
2. Estructuras de control
3. Funciones
4. Manejo de la memoria
5. Bibliotecas
6. Módulos

Historia de C

- BCPL (1967), B (1970), C (1972), ANSI-C (1988), C++ ...
- Conciso, pequeño
- Programación de sistemas
  - Bajo nivel, poco tipado, inseguro
- Ligado a UNIX desde 1973
- ANSI C 1989 - ISO C 1990

Un programa en C

```
/* vacio.c */
```

comentarios

```
#define N 100000000
```

directivas de preproceso

```
void vacio (void) { }
```

definición de funciones

```
int main (void ) {
    long i;
    for (i = 0; i < N; i ++)
        vacio ();
    exit (0);
}
```

función main

Tipos de datos

- Fundamentales:
  - enteros
  - reales
- Derivados:
  - enumerados
  - array
  - puntero
  - estructura
  - unión
  - campo de bits

Tipo nulo

- void
- el tipo que representa la nada
  - el tipo que devuelve un procedimiento
  - el tipo de los parámetros de las funciones sin parámetros
- el tipo “indefinido”
  - referencias a memoria de la que no se conoce el tipo

Tipos enteros

- char c = ´s´;
  - entero de 8 bits: -128, 127
    - Suele representar caracteres: ´x´ 64 -3 ´\n´ ´\t´ ´\033´
- int i = 25, j = 33;
  - entero de 16 o 32 bits: 124 0377 0x1af
- long l=25;
  - entero de 32 o 64 bits
- short s = 25;
  - entero menor o igual a int

Modificadores enteros

- Signo
  - unsigned
    - sólo positivos
    - unsigned i= 25;
    - unsigned long l= 25;
  - signed
- Tamaño
  - long
  - short

sizeof(char) = 1

sizeof (short) <= sizeof (int) <= sizeof (long)

Tipos reales

- float f = 0.56;
  - coma flotante de precisión simple en 32 bits
  - 124.5 3e-2
- double d= 0.56;
  - coma flotante precisión doble en 64 bits
- long double ld= 0.56;
  - coma flotante precisión doble en 80 bits

Conversiones

- Automáticas
  - promoción al de más rango
    - int + float = float
    - int = char;
    - int = short;
  - pérdida de información
    - char = int;
    - int = double;
- Manuales (cast)
  - (double) integer

Operadores

- Aritméticos

  + - * / %

- Relacionales y lógicos

  > >= < <= == != && || !

- Manejo de bits

  & | ^ << >> ~

Asignaciones

- Son operadores (devuelven un valor)

```
a = b + 1;
x = y = z = 3;
u = (v = x) + (y = z);
```

- Combinables con otros operadores

```
i += 2;
z *= y - 1;
mask <<= 3;
```

Otros operadores

- Incremento/decremento

  i ++; a [++i] --

- Evaluación condicional

  (a > b) ? a : b;

- Tamaño en bytes

  sizeof (variable); sizeof (tipo);

- Secuencia

  i = (j ++ , 2 * j);

Expresiones

- Variables, constantes, llamadas y operadores para obtener un valor tipado

```
(vueltas / tics) * clocks_per_tic / 500000
```

- conversiones automáticas
- expresiones lógicas y de relación -> enteros
  - 0 es falso, el resto cierto

```
if (! --vueltas) exit(0);
```

Evaluación de expresiones

Orden de evaluación

- paréntesis
- reglas de precedencia y agrupación de los operadores
- no se especifica en qué orden se evalúan los operandos de un operador

```
x = f() + g();  /* f y g, o g y f ? */
```

- los operadores && y || evalúan lo mínimo posible

Precedencia y agrupación

Operador	Agrupación
() [] -> .	->
! ~ ++ -- - (cast) * & sizeof	<-
* / % + - << >> < <= > >= == != & ^ && ||	->
?: = += -=	<-
,	->

Bloques

- Sentencias simples: acaban en ;

```
int i;
```

```
int tics = clock ();
```

```
for (i = vueltas; i > 0; i -- ) ;
```

- Sentencia compuesta: bloque {}

```
if (ticks >= CLOCK_PER_SEC)
{
  printf("Son %5.2f BogoMips\n",
          (vueltas / tics)*CLOCKS_PER_SEC/ 500000);
  exit (0);
}
```

Condicionales

- Si una condición es cierta (distinto de 0)
  - se ejecuta la rama if
  - si no, la rama else (si existe)

```
if (pos == vacio) {
    escribir_registro(das, pos, pc, pv);
    puesto = 0;
}
else if (pos  == borrado) {
    escribir_registro(das, pos, pc, pv);
    puesto = 1;
}
```

Decisión múltiple

```
switch ( c ) {          /* expresión */
    case 's':           /* salta al siguiente */
    case 'S':   si ();
                break;
    case 'n':
    case 'N':   no();
                break;
    default:    error ();
}
```

Bucles while-do y do-while

```
while ( ! cansado ) {
    trabaja ();
}
```

```
do {
   trabaja ();
} while (! cansado);
```

Bucle for

```
for (n = 0, j = 10 ;
     n < 100  ;
     n++ , j--) {
  HacerCosas(n,j);
}
```

```
for ( i = vueltas ; i > 0 ; i -- )
   ;
```

```
for ( ;  ; ) {
  ...
  break;  ... continue;
}
```

El entorno de programación

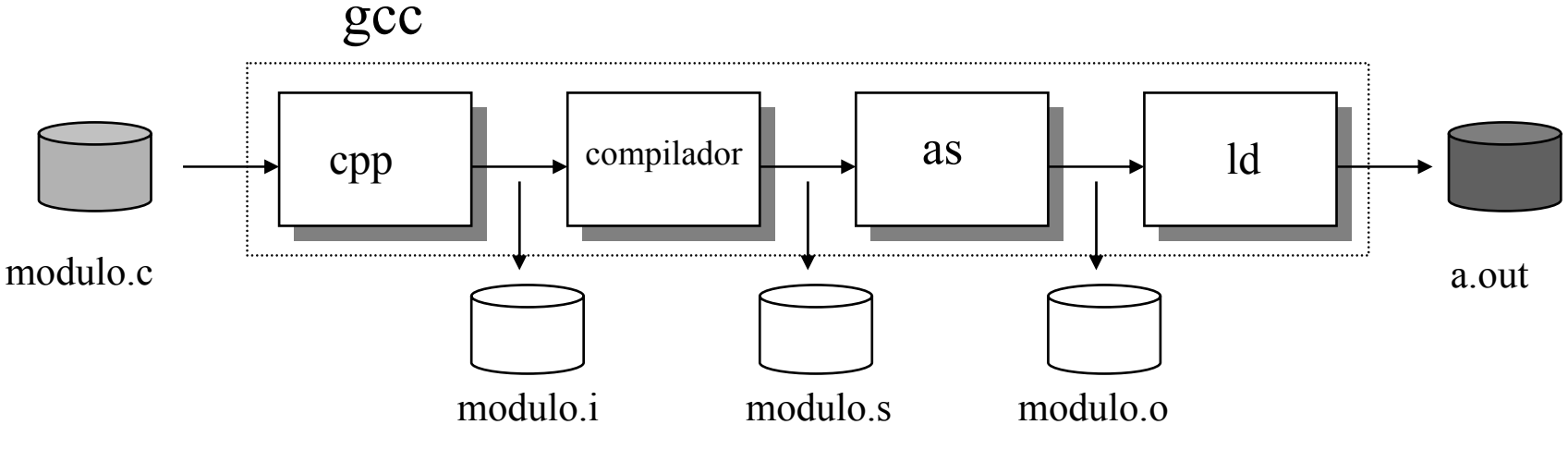
Entorno de programación

- Conjunto de herramientas
  que pueden trabajar coordinadamente
  para facilitar la labor del programador
- Mecanismo de conexión: sistema de ficheros
- Mínimo:
  - editor (joe, vi, emacs)
  - compilador (gcc)
  - montador de enlaces (ld, gcc)

Entornos

- entorno avanzado
  - entorno mínimo +
  - gestión de bibliotecas (ar, ranlib, nm)
  - gestor de dependencias y constructor (make)
  - depuradores (gdb, strace)
  - medidor de rendimiento (gprof)
  - comprobador de accesos a memoria (efence)
- entorno integrado:
  - todas las herramientas con interfaz común
  - coordinación entre ellas

Fases del compilador



gcc -opciones ficheros

Llamada al compilador

- Opciones:
  - control de salida, preprocesador, ensamblador, montador
    $ gcc -c -o ejecutable -v programa.c
  - elementos del lenguaje, avisos
    $ gcc -ansi -Wall programa.c
  - optimización
    $ gcc -O3 programa.c
  - definición de nombres
    $ gcc -DSO=LINUX programa.c

Ejemplo

```
/* Formula C= (5/9)(F-32) */
int main (void) {
  int desde, hasta, paso;
  float fahr, celsius;

  desde=   0; /* límite inferior */
  hasta= 300; /* límite superior */
  paso= 20;

  fahr= desde;
  while (fahr <= hasta) {
    celsius= (5.0/9.0) * (fahr - 32.0);
    printf("%f %f\n", fahr, celsius);
    fahr= fahr + paso;
  }
  exit(0);
}
```