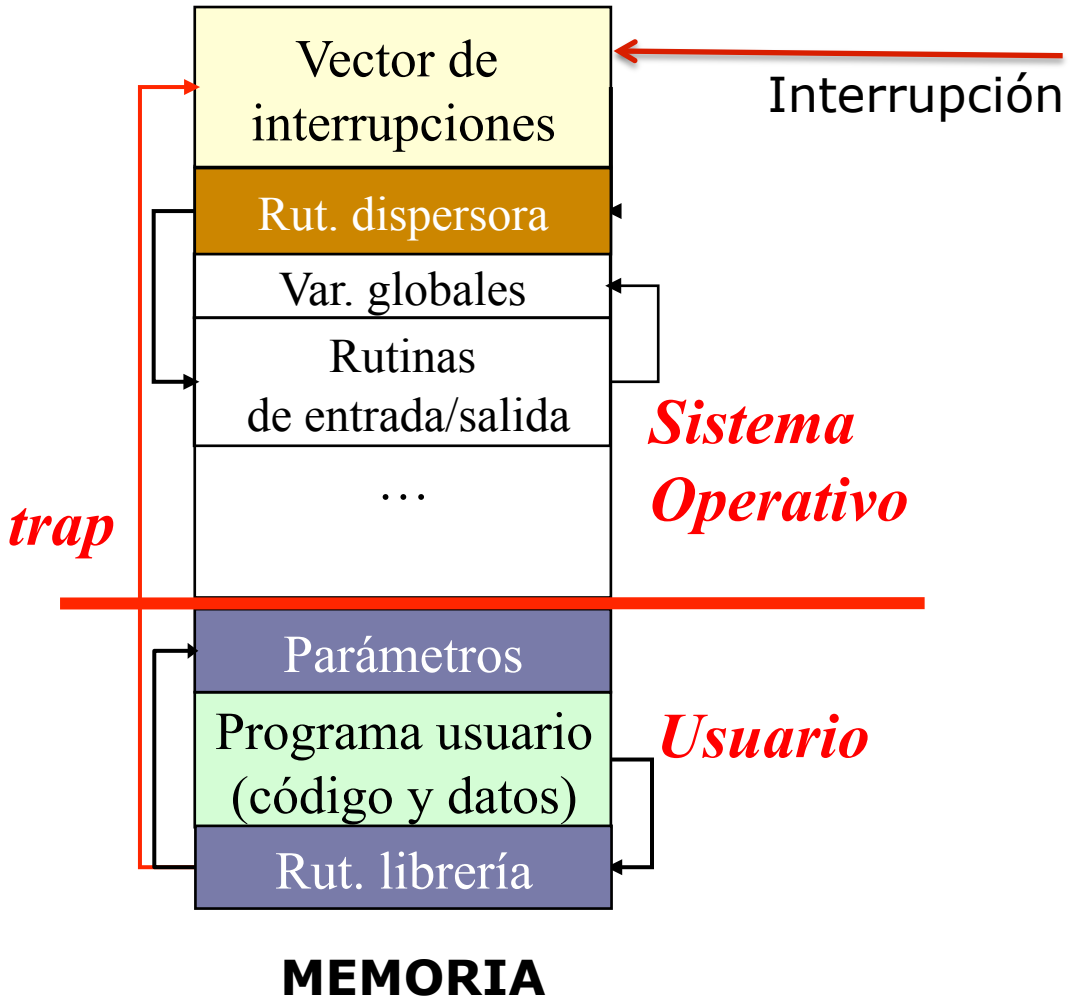
Tema 3
Entrada/salida

Independencia del dispositivo

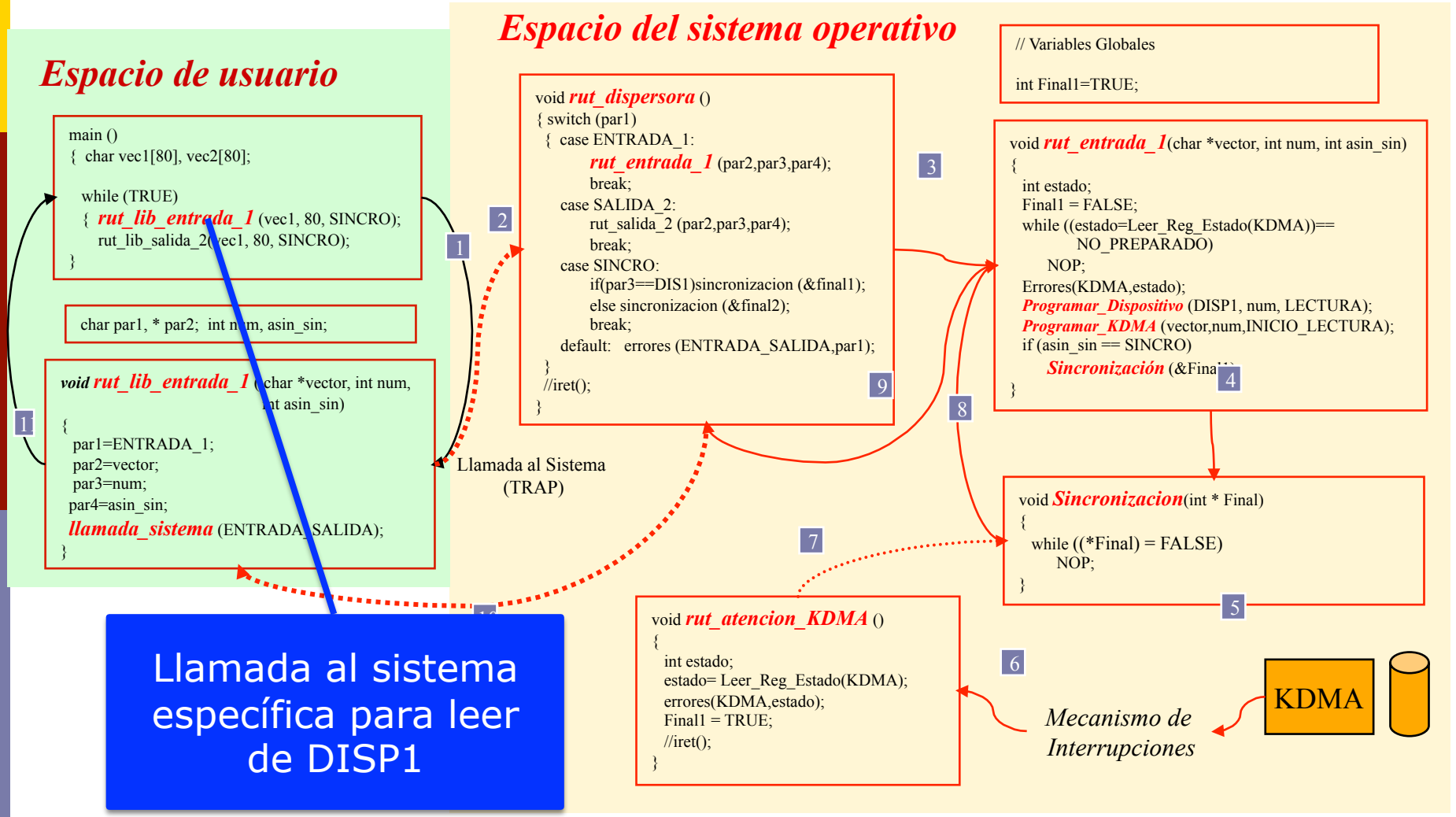
Qué vamos a aprender

Cómo programar aplicaciones sin pensar en qué dispositivos de entrada/salida usan

Acceso al sistema operativo mediante llamadas al sistema



Ejemplo


Espacio de usuario
Espacio del sistema operativo
main ()
{ char vec1[80], vec2[80];
while (TRUE)
{ rut_lib_entrada_1 (vec1, 80, SINCRO);
rut_lib_salida_2(vec1, 80, SINCRO);
}
char par1, * par2; int num, asin_sin;
void rut_lib_entrada_1 (char *vector, int num, int asin_sin)
{
par1=ENTRADA_1;
par2=vector;
par3=num;
par4=asin_sin;
llamada_sistema (ENTRADA_SALIDA);
}
void rut_dispersora ()
{ switch (par1)
{ case ENTRADA_1:
rut_entrada_1 (par2,par3,par4);
break;
case SALIDA_2:
rut_salida_2 (par2,par3,par4);
break;
case SINCRO:
if(par3==DIS1)sincronizacion (&final1);
else sincronizacion (&final2);
break;
default: errores (ENTRADA_SALIDA,par1);
}
//iret();
}
// Variables Globales
int Final1=TRUE;
void rut_entrada_1(char *vector, int num, int asin_sin)
{
int estado;
Final1 = FALSE;
while ((estado=Leer_Reg_Estado(KDMA))==
NO_PREPARADO)
NOP;
Errores(KDMA,estado);
Programar_Dispositivo (DISP1, num, LECTURA);
Programar_KDMA (vector,num,INICIO_LECTURA);
if (asin_sin == SINCRO)
Sincronización (&Final1);
}
void Sincronizacion(int * Final)
{
while ((*Final) = FALSE)
NOP;
}
void rut_atencion_KDMA ()
{
int estado;
estado= Leer_Reg_Estado(KDMA);
errores(KDMA,estado);
Final1 = TRUE;
//iret();
}
Mecanismo de Interrupciones
KDMA
Llamada al Sistema (TRAP)
1
2
3
4
5
6
7
8
9
Llamada al sistema específica para leer de DISP1

Independencia del Dispositivo

- Según el esquema anterior:
  - Un dispositivo → Una llamada al sistema específica
  - Cambiar de dispositivo → Modificar el programa
- ¿Cómo independizar el programa de los dispositivos utilizados?
  - ➢ El **canal** como abstracción del dispositivo
  - En UNIX: *file descriptor*

Independencia del dispositivo

Rutinas de biblioteca

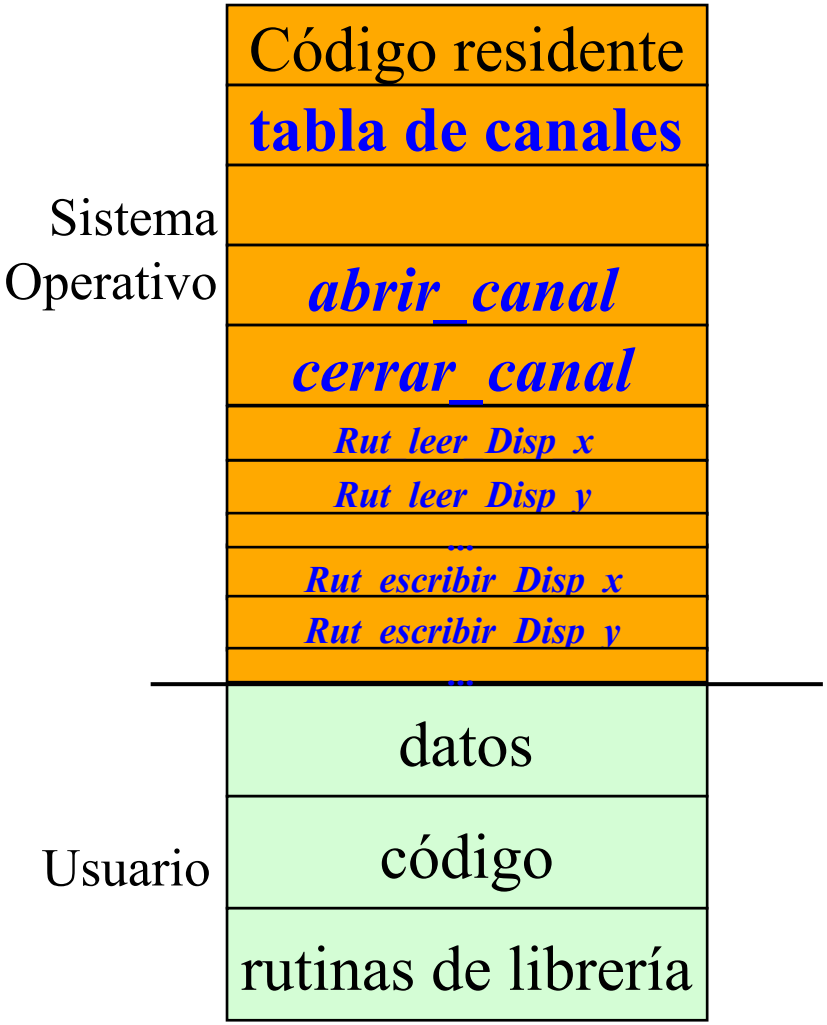
```
void rut_lib_leer (int canal char *vector,
                   int num, int asin_sin)
{
  par0=canal;
  par1=LEER;
  par2=vector;
  par3=num;
  par4=asin_sin;
  llamada_sistema (ENTRADA_SALIDA);
}
```

```
void rut_lib_escribir (int canal char *vector,
                   int num, int asin_sin)
{
  par0=canal;
  par1=ESCRIBIR;
  par2=vector;
  par3=num;
  par4=asin_sin;
  llamada_sistema (ENTRADA_SALIDA);
}
```

Programa usuario (síncrono)

```
main ()
{
  int f1, f2;
  char vec1[80], vec2[80];
  rut_lib_abrir(f1,"dispositivo_entrada" );
  rut_lib_abrir(f2,"dispositivo_salida");
  while (TRUE) {
     rut_lib_leer (f1,vec1,80,SINCRO);
     rut_lib_escribir (f2,vec1,80,SINCRO);
   }
  rut_lib_cerrar (f1);
  rut_lib_cerrar (f2);
}
```

Independencia del Dispositivo



	direcciones de rutinas de E/S			
canal	LEER	ESCRIBIR	POSICIONAR	
1	@x	@y	@z	
2	@w			
3				

Tabla de canales