Programación Concurrente y Tiempo Real

Laboratorio - Práctica 2
Semáforos y Memoria Compartida

D. Vallejo, M.A. Redondo, J. Albusac,
C. González, J. Ruiz

Escuela Superior de Informática

Universidad de Castilla-La Mancha

Práctica 2. Semáforos y M.C.

1. Introducción
2. Primitivas
3. Análisis del Problema
4. Coordinación de Procesos
5. Tipos de Datos
6. Gestión de Memoria Compartida

Introducción

- IPC POSIX: Estándar en 1993 (Real Time).
- Existen otros IPC en UNIX como SystemV.

Introducción

- IPC POSIX: Estándar en 1993 (Real Time).
- Existen otros IPC en UNIX como SystemV.

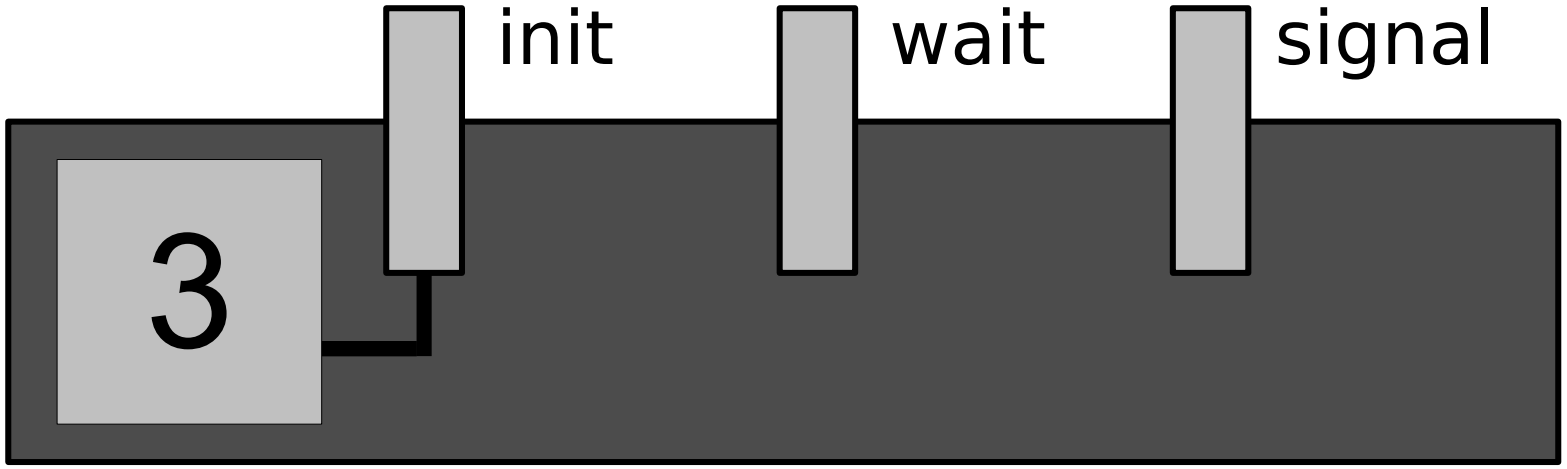
Generalidades POSIX

- Familia de estándares de llamadas al sistema definidos por IEEE 1003.
- Uso de nombrado de Objetos.
- Mecanismo más sencillo que System V.
- Gran compatibilidad en UNIX.
- Posible uso en otros SSOO (Windows).

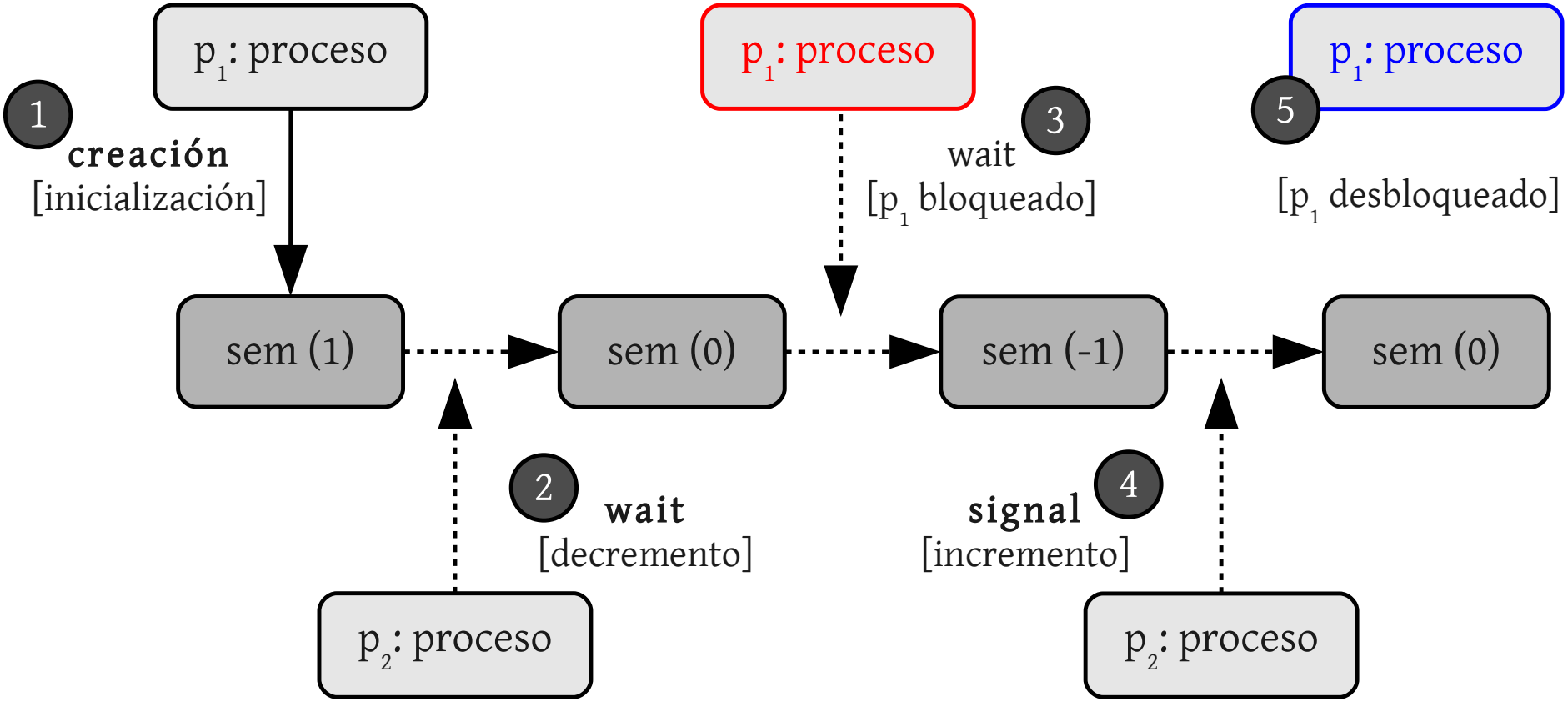
Semáforos

Caja negra

TAD con 3 Operaciones



Semáforos


p1: proceso
1
creación
[inicialización]
sem (1)
sem (0)
p1: proceso
3
wait
[p1 bloqueado]
sem (-1)
p1: proceso
5
[p1 desbloqueado]
sem (0)
2
wait
[decremento]
p2: proceso
4
signal
[incremento]
p2: proceso

Semáforos nombrados en POSIX

Manipulación de semáforos

```
#include <semaphore.h>

/* Devuelve un puntero al semáforo o SEM FAILED */
sem_t *sem_open(
    const char *name,  /* Nombre del semáforo */
    int oflag,         /* Flags               */
    mode_t mode,       /* Permisos            */
    unsigned int value /* Valor inicial       */
 );

int sem_close  (sem_t *sem);
int sem_unlink (const char *name);

int sem_wait (sem_t *sem);
int sem_post (sem_t *sem);
```

Semáforos nombrados en POSIX

TAD semáforo

```
#ifndef __SEMAPHOREI_H__
#define __SEMAPHOREI_H__

#include <semaphore.h>

sem_t *create_semaphore (const char *name,
                         unsigned int value);
sem_t *get_semaphore    (const char *name);
void remove_semaphore   (const char *name);
void signal_semaphore   (sem_t *sem);
void wait_semaphore     (sem_t *sem);

#endif
```

Semáforos nombrados en POSIX

create_semaphore

```
#include <semaphoreI.h>

sem_t *create_semaphore (const char *name,
                         unsigned int value) {
  sem_t *sem;

  if ((sem = sem_open(name, O_CREAT, 0644, value))
       == SEM_FAILED) {
    fprintf(stderr, "Error creating semaphore <%s>: %s\n",
            name, strerror(errno));
    exit(EXIT_FAILURE);
  }

  return sem;
}
```

Memoria compartida en POSIX

Gestión de segmentos de memoria

```
#include <mman.h>
#include <unistd.h>

/* Devuelve el descriptor de archivo o -1 si error */
int shm_open(
  const char *name, /* Nombre del segmento */
  int oflag,        /* Flags               */
  mode_t mode       /* Permisos            */
);

int shm_unlink (const char *name);
int close      (int fd);
int ftruncate  (int fd, off_t length);
```

Memoria compartida en POSIX

Mapping

```
#include <sys/mman.h>

void *mmap(
  void *addr,     /* Dirección de memoria  */
  size_t length,  /* Longitud del segmento */
  int prot,       /* Protección            */
  int flags,      /* Flags                 */
  int fd,         /* Descriptor de archivo */
  off_t offset    /* Desplazamiento        */
);

int munmap(
  void *addr,     /* Dirección de memoria  */
  size_t length   /* Longitud del segmento */
);
```

Estructura de datos compartida

struct TTask_t

```
struct TTask_t { int begin; int end; };

int shm_task; struct TTask_t *task;

/* El gestor crea el segmento de memoria compartida */
shm_task = shm_open(SHM_TASK, O_CREAT | O_RDWR, 0644);
ftruncate(shm_task, sizeof(struct TTask_t));
task = mmap(NULL, sizeof(struct TTask_t),
            PROT_READ | PROT_WRITE, MAP_SHARED,
            shm_task, 0);
/* Manejo de la v.m.c */
task->begin = 0; task->end = 10;

close(shm_task);

/* El gestor libera el segmento de memoria compartida */
shm_unlink(SHM_TASK);
```

Contexto

- Problema tipo Prueba de Laboratorio.
- Se suministra plantilla de código fuente.
- 70 Minutos.
- No se permiten libros o apuntes.
- Test de resultado determinista.
- Competencias a Evaluar:
  - Resolución de Problema de Concurrencia.
  - Dominio de llamadas POSIX.
  - Construcción automática con Make.
  - Dominio del Entorno de Trabajo.

Análisis del Problema de Ejemplo

Traducción Concurrente

Entero	**1**	**2**	**...**	**25**	**26**	**27**	**28**	**...**	**51**	**52**	**53**	**54**	**55**	**56**	**57**
Traducción	**a**	**b**	**...**	**y**	**z**	**A**	**B**	**...**	**Y**	**Z**	**.**	**,**	**!**	**?**	**_**
ASCII	*97*	*98*	*...*	*121*	*122*	*65*	*66*	*...*	*89*	*90*	*46*	*44*	*33*	*63*	*95*

Análisis del Problema de Ejemplo

Traducción Concurrente

Entero	1	2	...	25	26	27	28	...	51	52	53	54	55	56	57
Traducción	a	b	...	y	z	A	B	...	Y	Z	.	,	!	?	_
ASCII	*97*	*98*	*...*	*121*	*122*	*65*	*66*	*...*	*89*	*90*	*46*	*44*	*33*	*63*	*95*

Ejemplo:

“26.27.2.1.56” → TRADUCCIÓN → 122 65 98 97 63 = “zZba?”

Entrada | Valores ASCII Traducidos | Caracteres

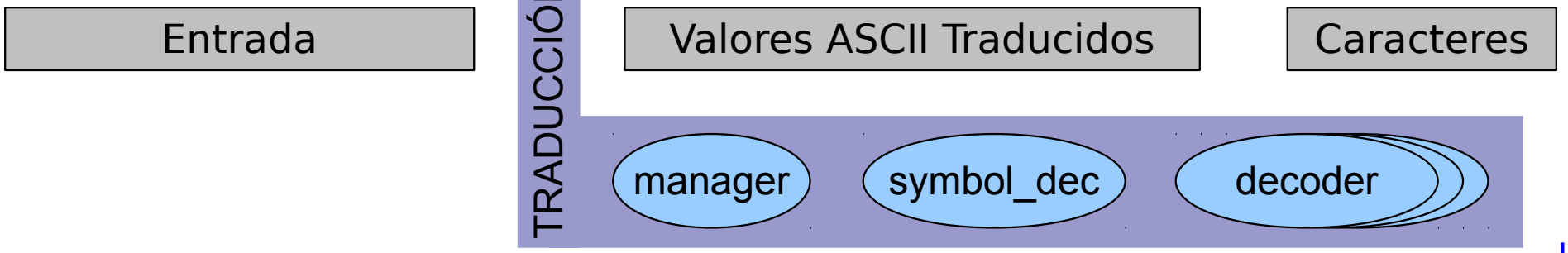
Análisis del Problema de Ejemplo

Traducción Concurrente

Entero	1	2	...	25	26	27	28	...	51	52	53	54	55	56	57
Traducción	a	b	...	y	z	A	B	...	Y	Z	.	,	!	?	_
ASCII	*97*	*98*	*...*	*121*	*122*	*65*	*66*	*...*	*89*	*90*	*46*	*44*	*33*	*63*	*95*

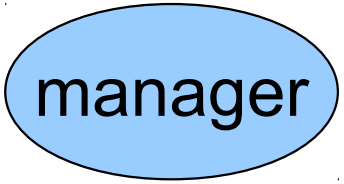
Ejemplo:

“26.27.2.1.56” → 122 65 98 97 63 = “zZba?”

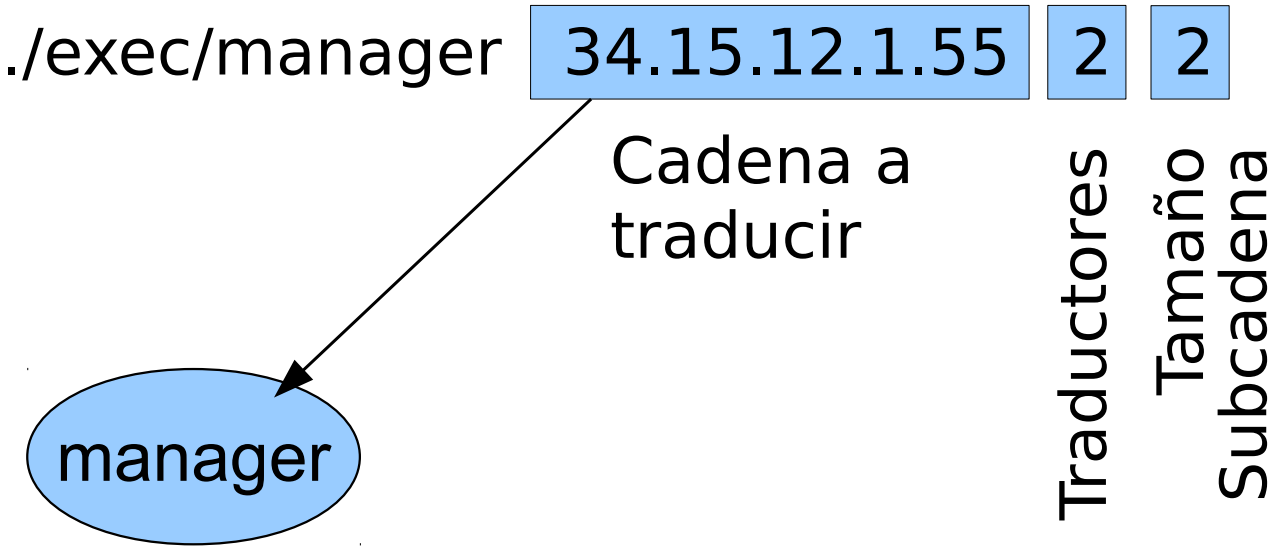


Análisis del Problema de Ejemplo

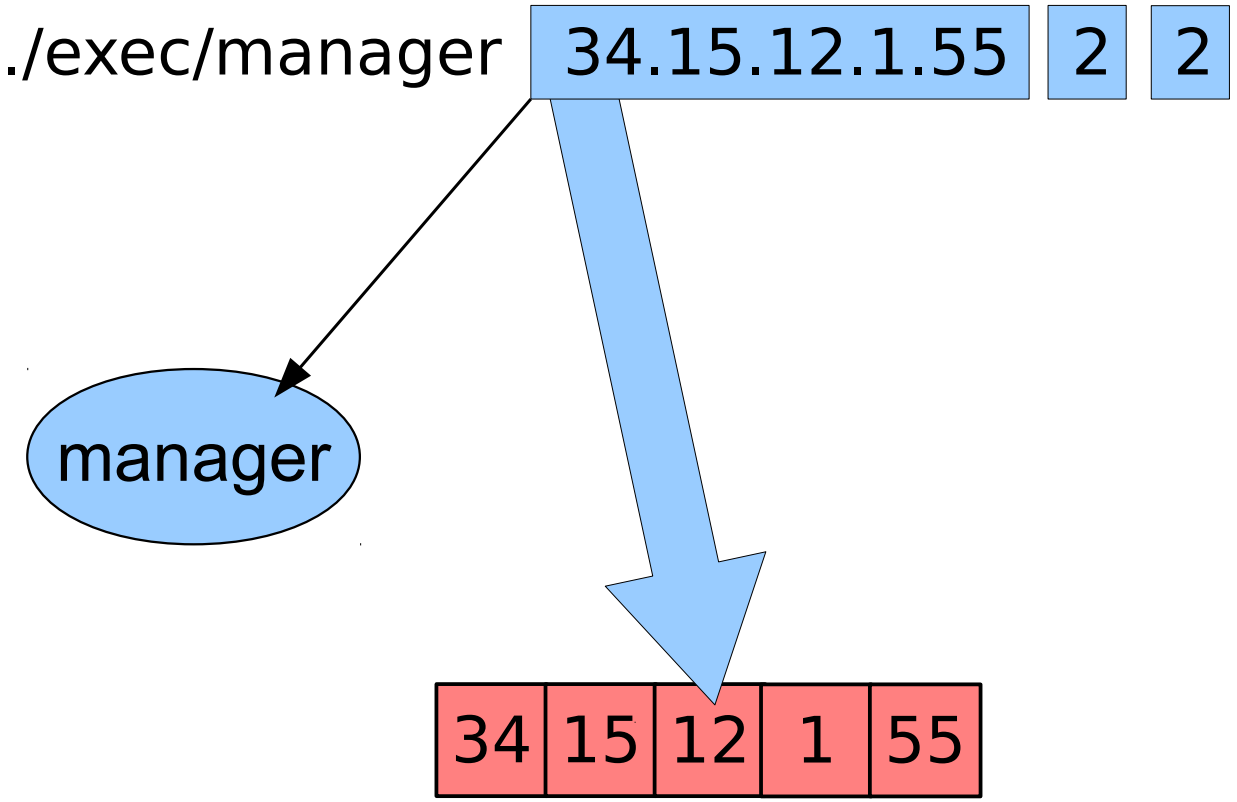
./exec/manager 34.15.12.1.55 2 2



Análisis del Problema de Ejemplo

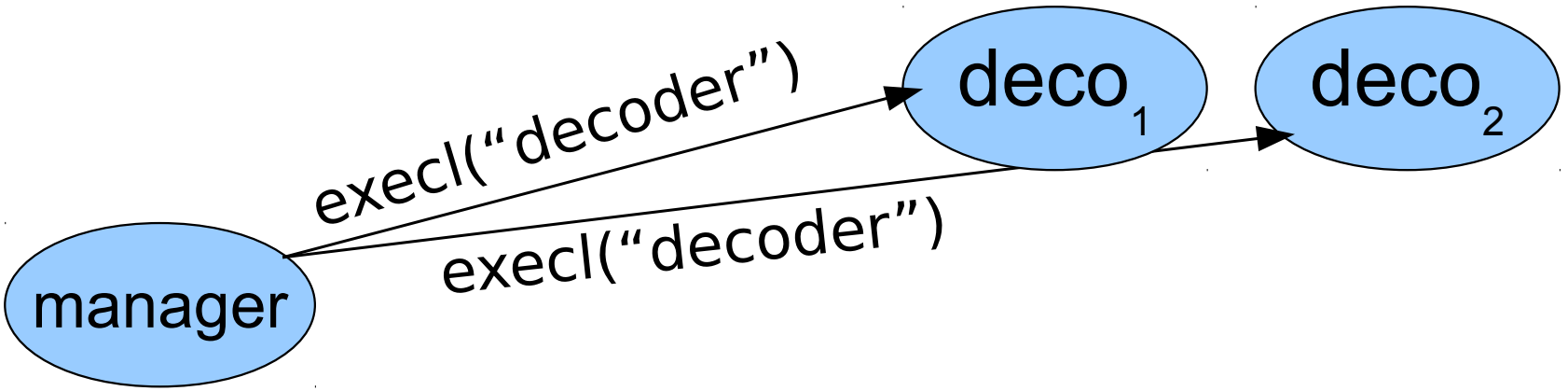


Análisis del Problema de Ejemplo



Análisis del Problema de Ejemplo

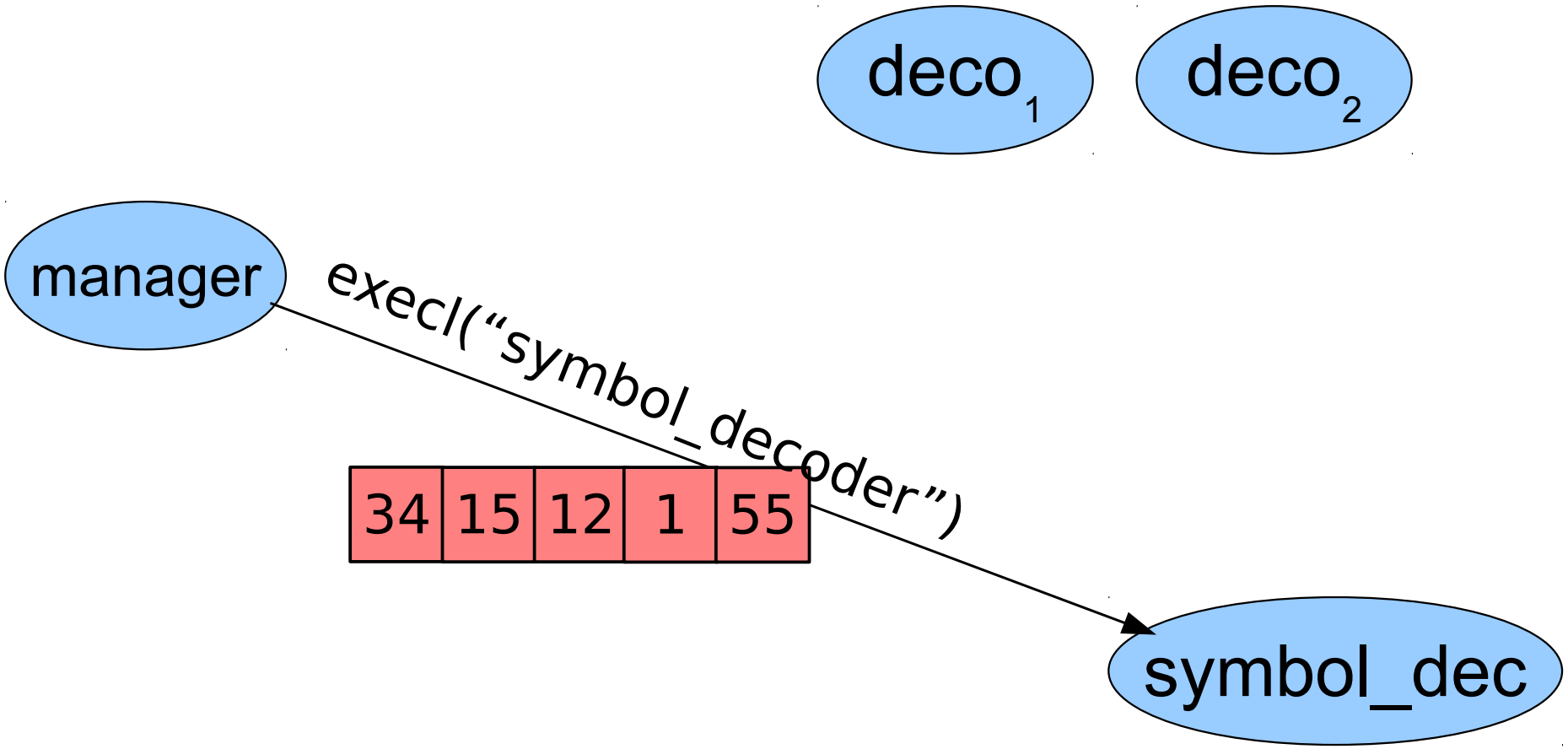
./exec/manager 34.15.12.1.55 2 2



34	15	12	1	55

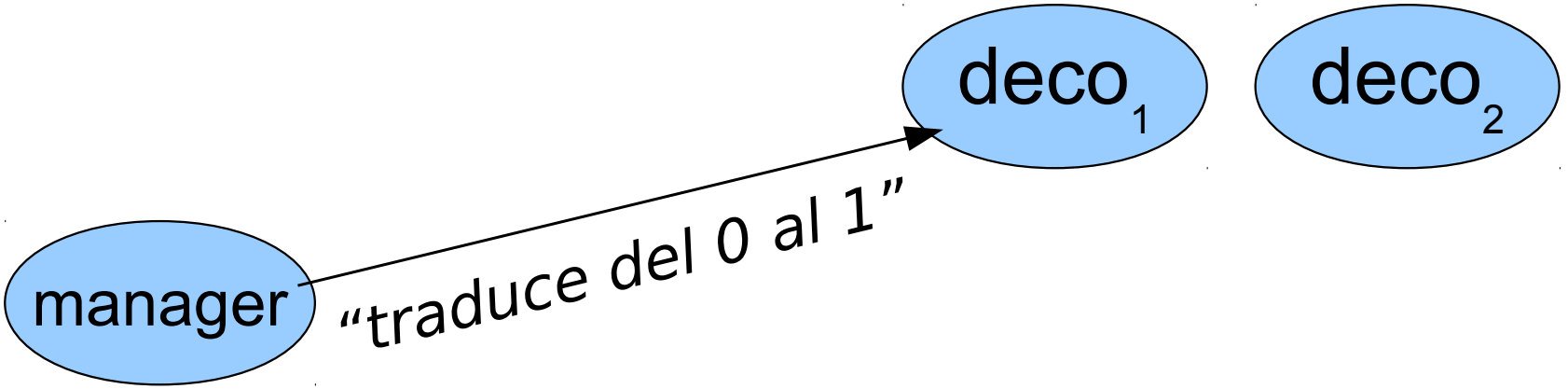
Análisis del Problema de Ejemplo

./exec/manager 34.15.12.1.55 2 2



Análisis del Problema de Ejemplo

./exec/manager 34.15.12.1.55 2 2

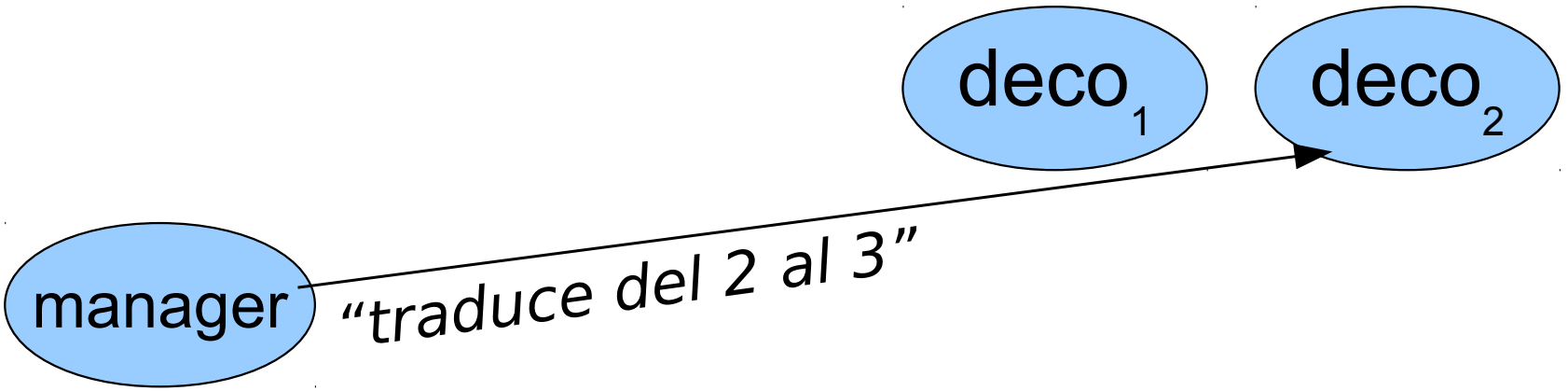


34	15	12	1	55

symbol_dec

Análisis del Problema de Ejemplo

./exec/manager 34.15.12.1.55 2 2

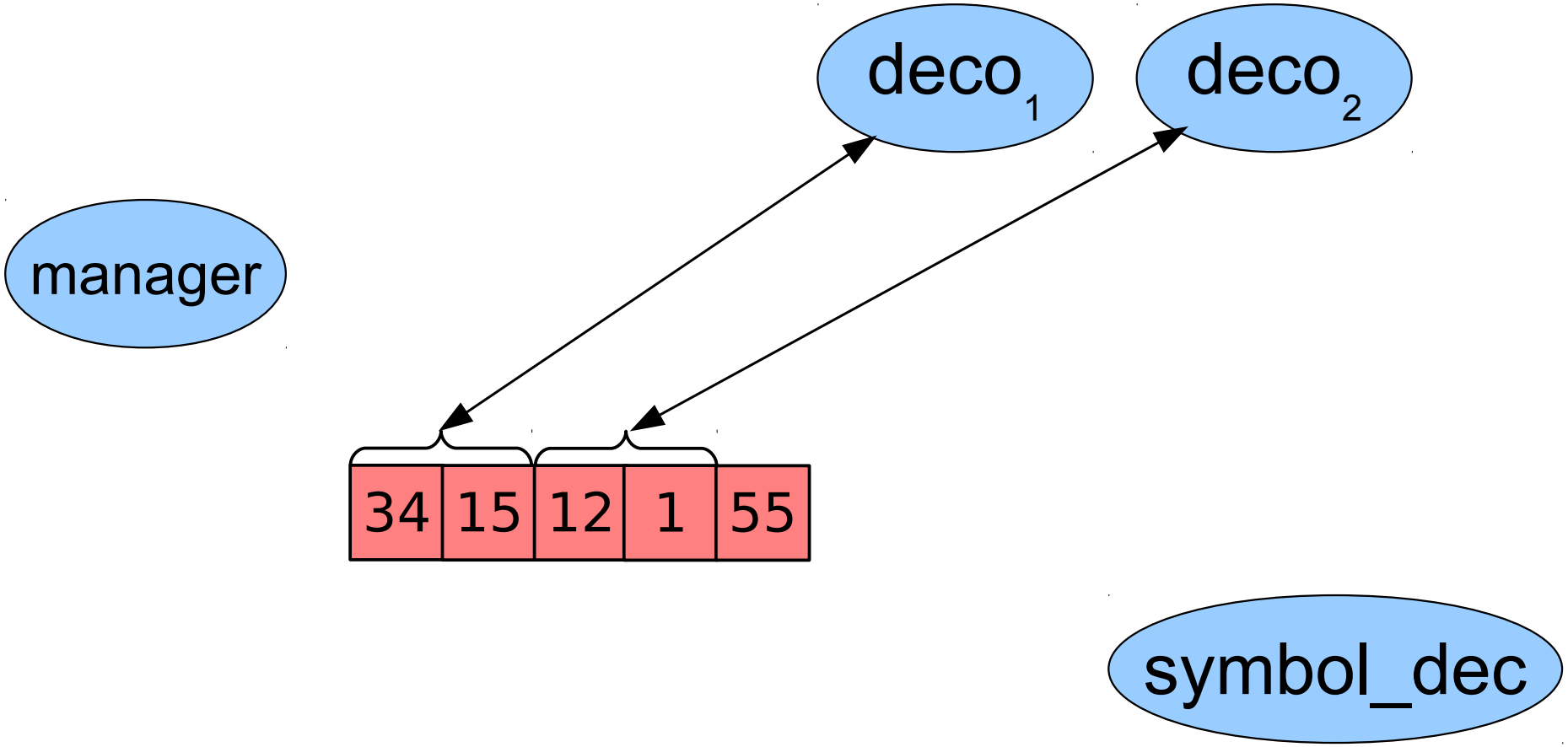


34	15	12	1	55

symbol_dec

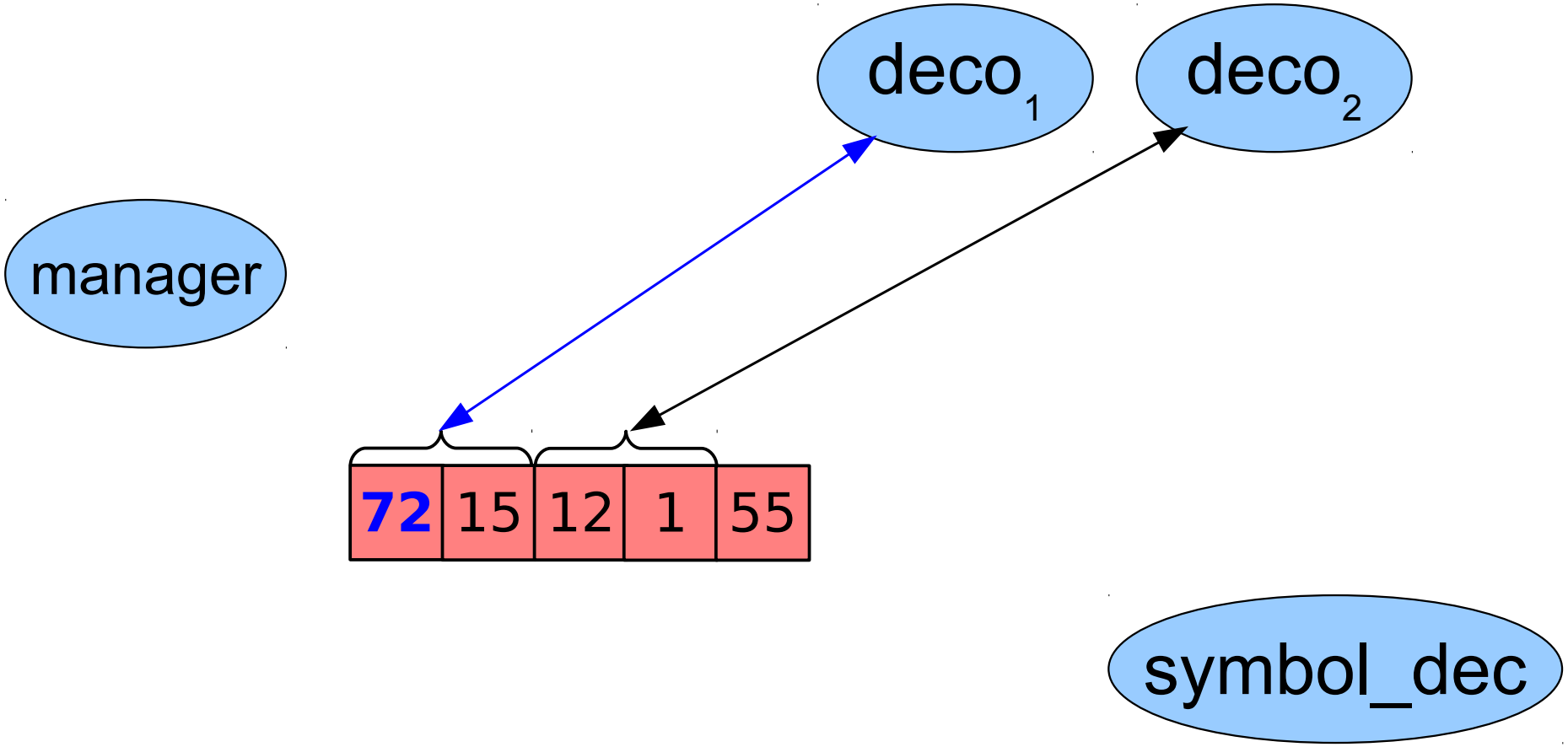
Análisis del Problema de Ejemplo

./exec/manager 34.15.12.1.55 2 2



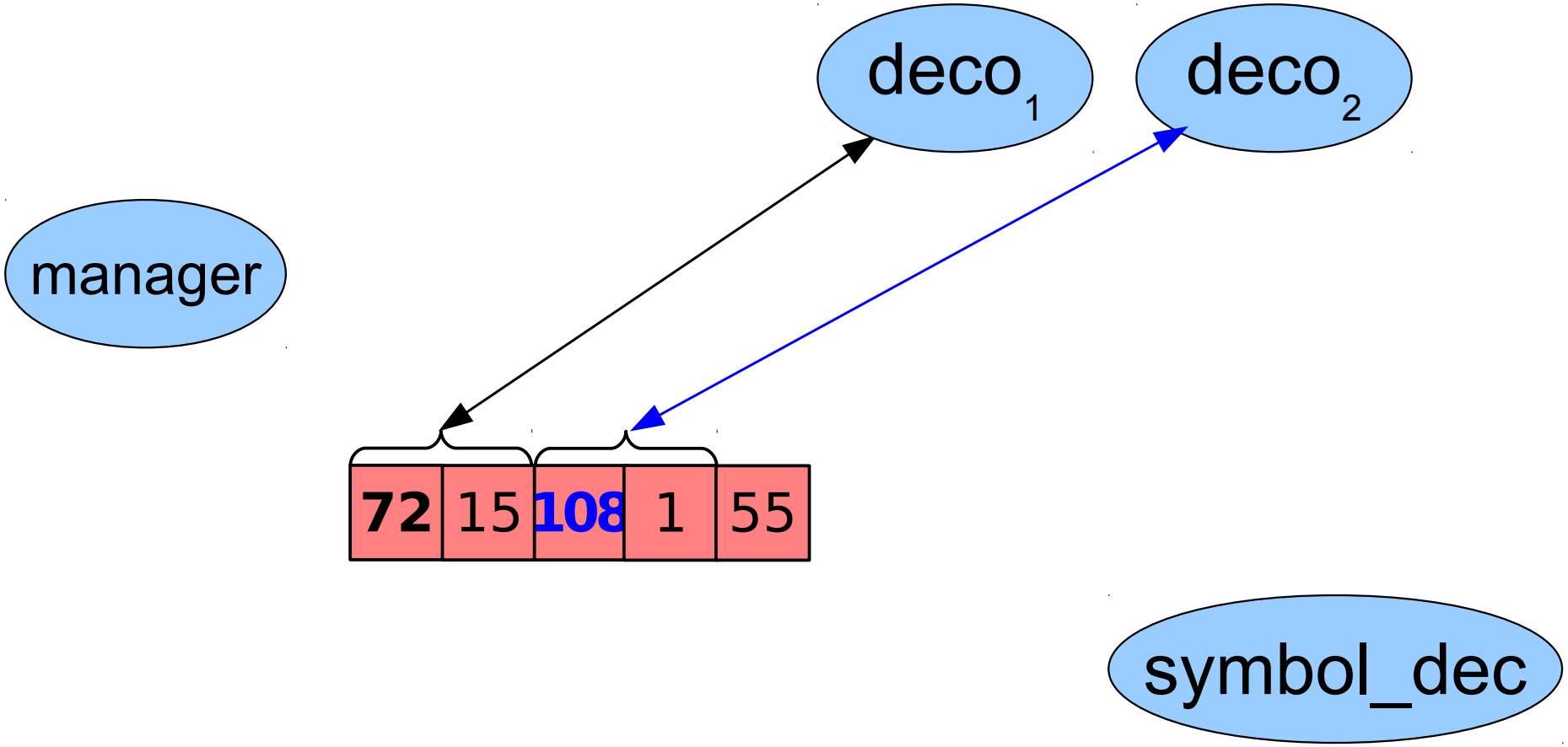
Análisis del Problema de Ejemplo

./exec/manager 34.15.12.1.55 2 2



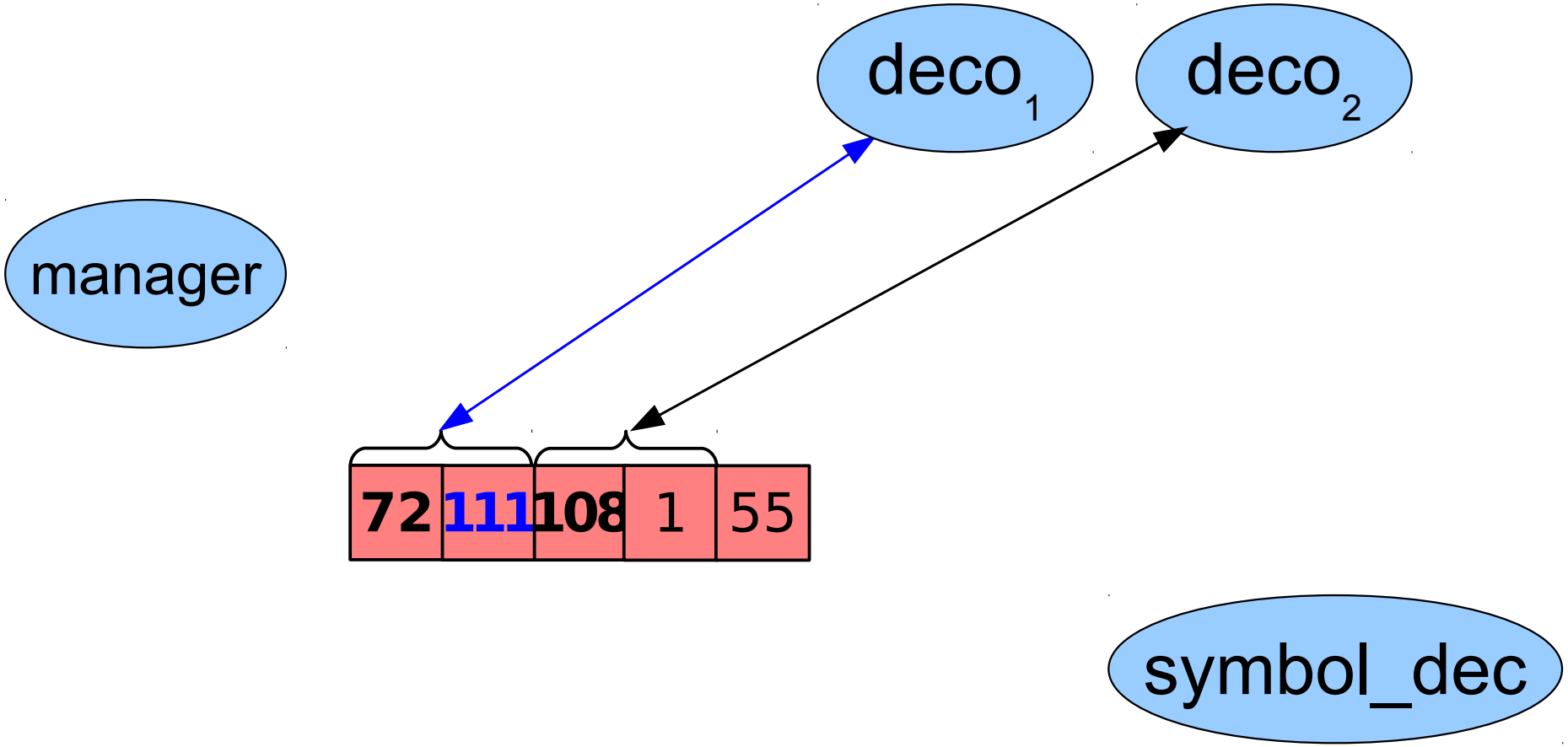
Análisis del Problema de Ejemplo

./exec/manager 34.15.12.1.55 2 2



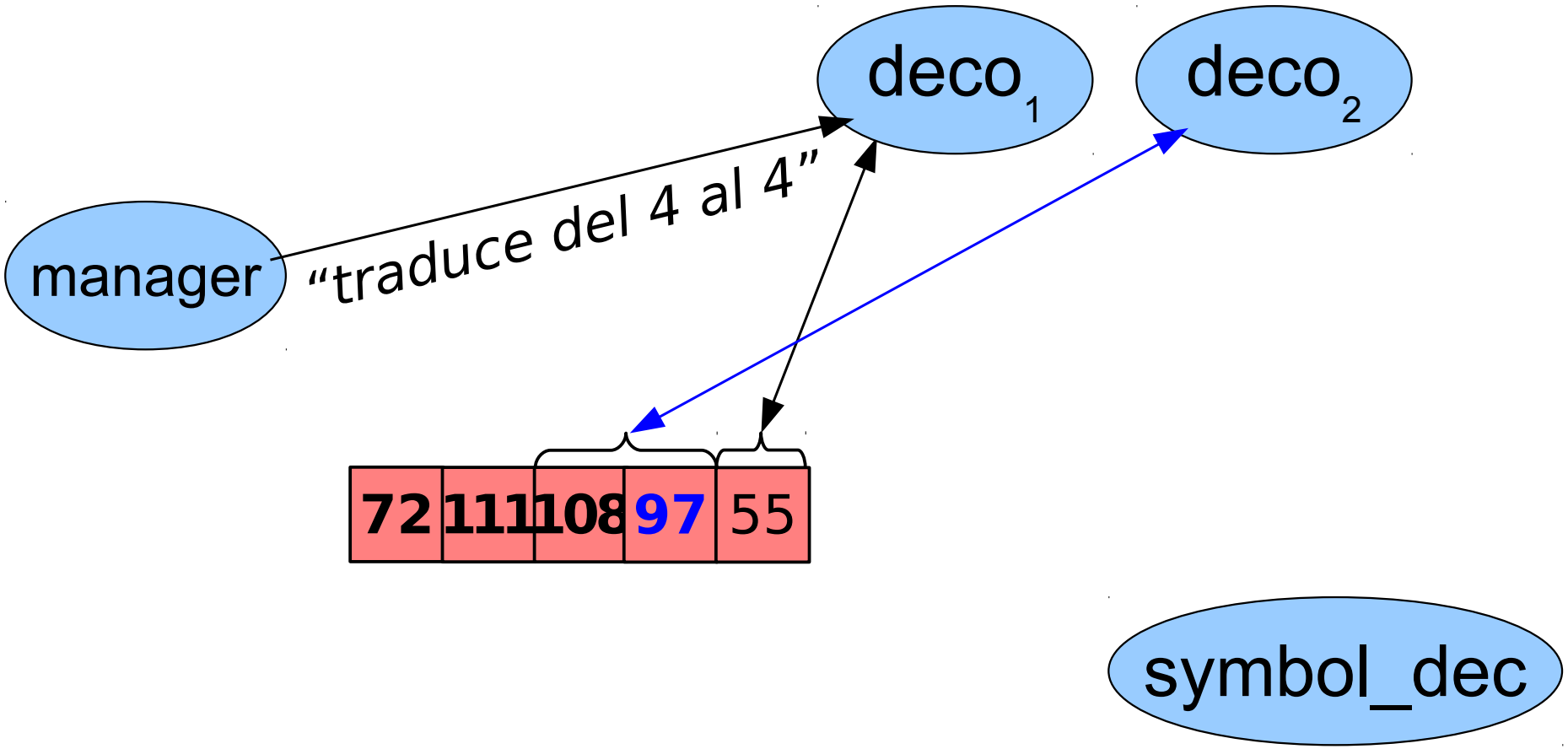
Análisis del Problema de Ejemplo

./exec/manager 34.15.12.1.55 2 2



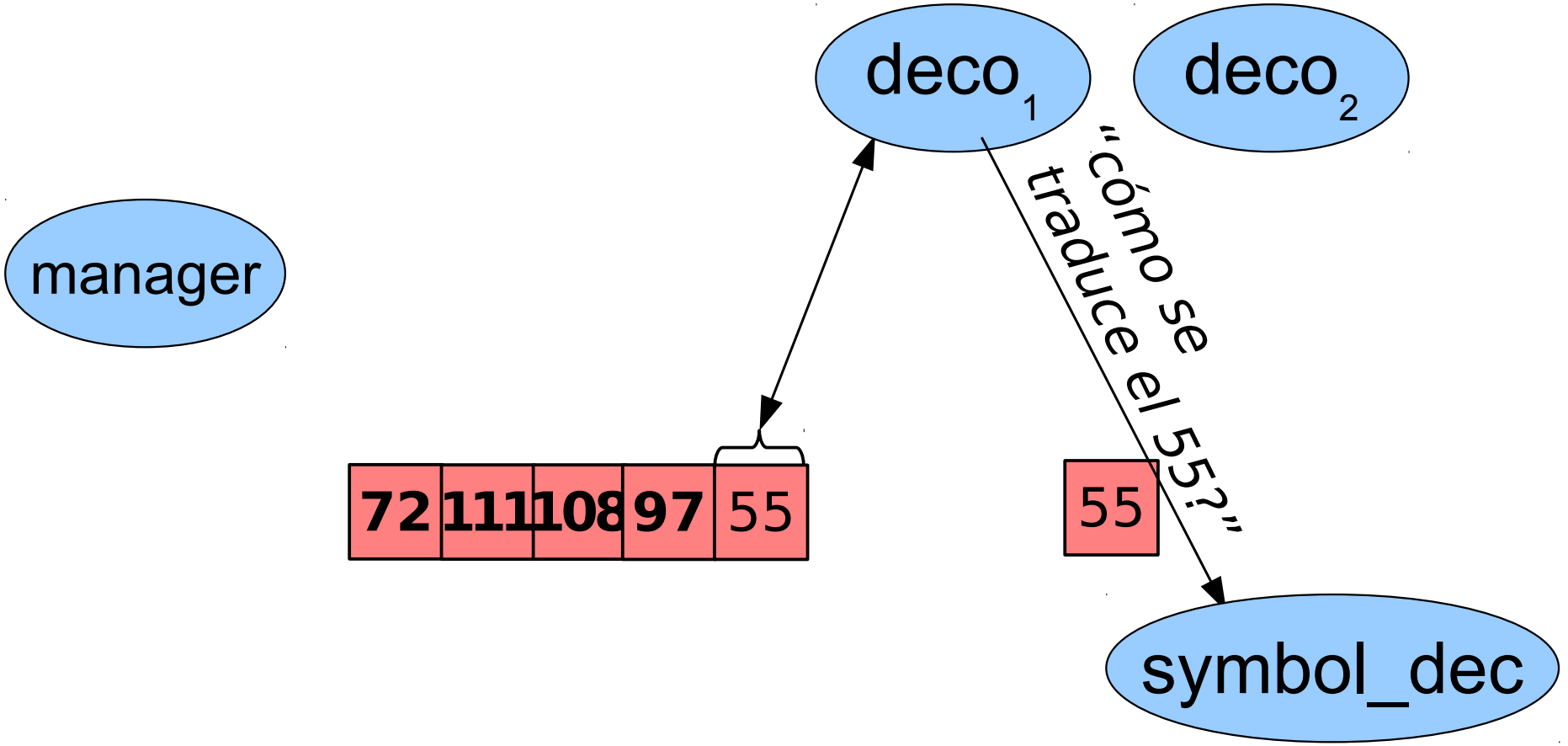
Análisis del Problema de Ejemplo

./exec/manager 34.15.12.1.55 2 2



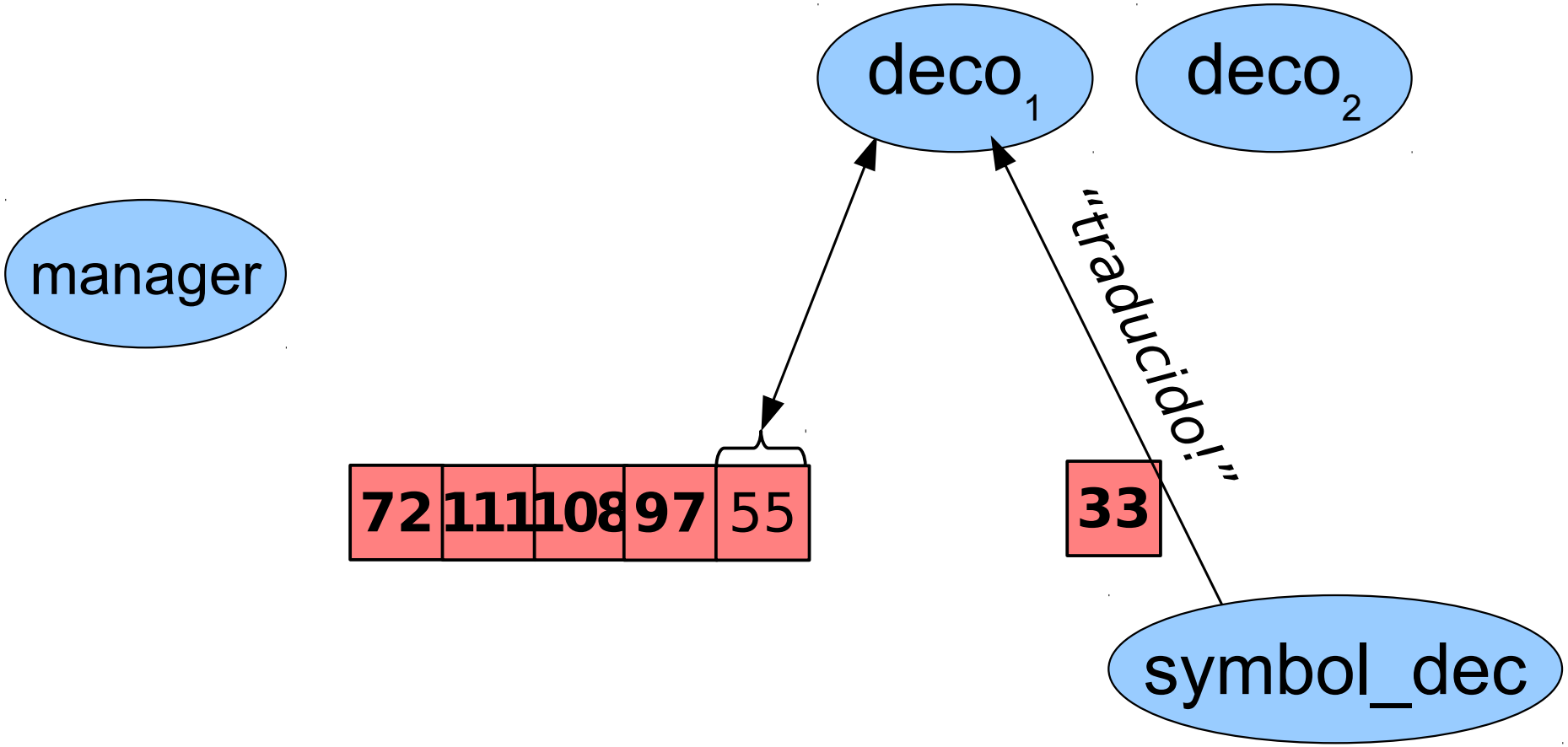
Análisis del Problema de Ejemplo

./exec/manager 34.15.12.1.55 2 2



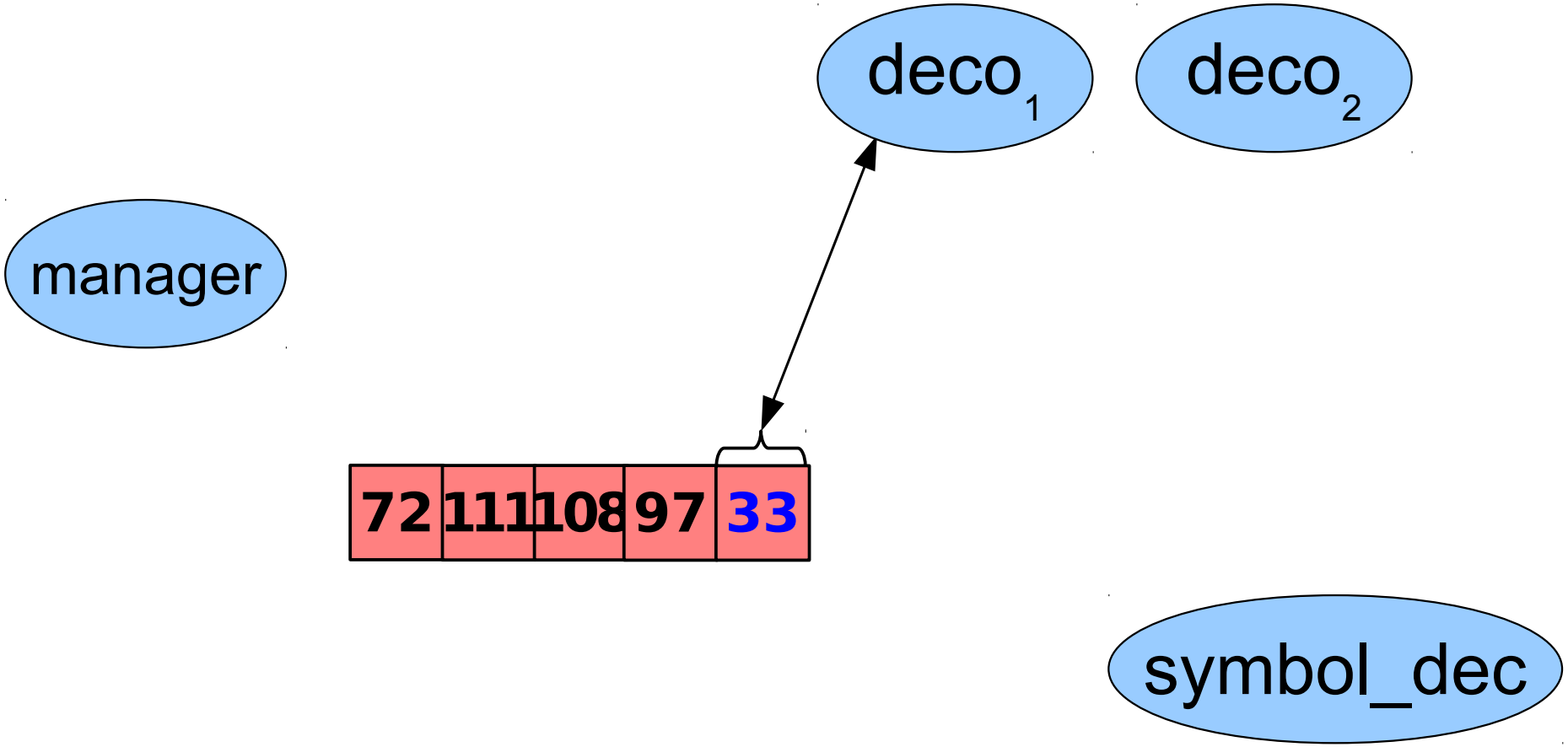
Análisis del Problema de Ejemplo

./exec/manager 34.15.12.1.55 2 2



Análisis del Problema de Ejemplo

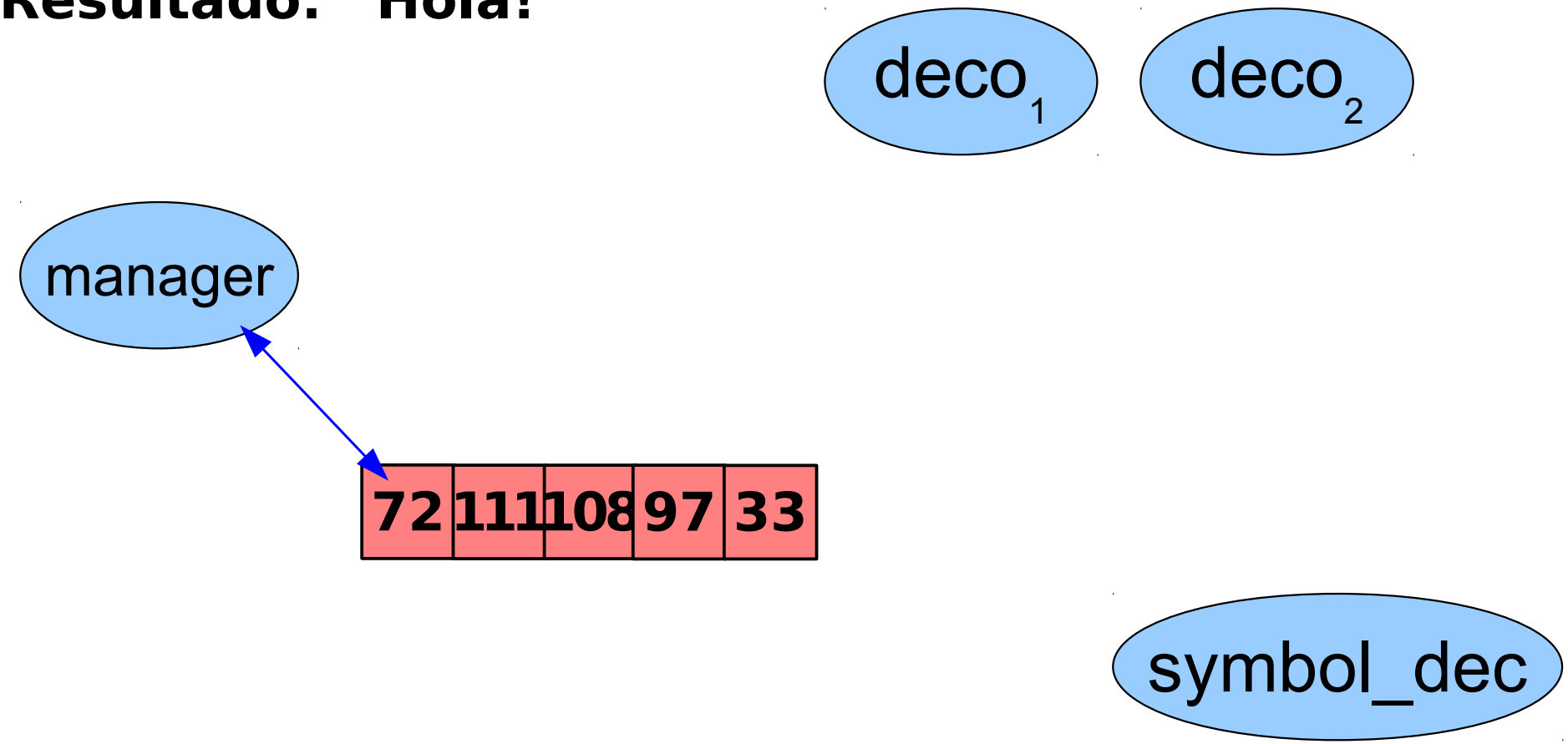
./exec/manager 34.15.12.1.55 2 2



Análisis del Problema de Ejemplo

./exec/manager 34.15.12.1.55 2 2

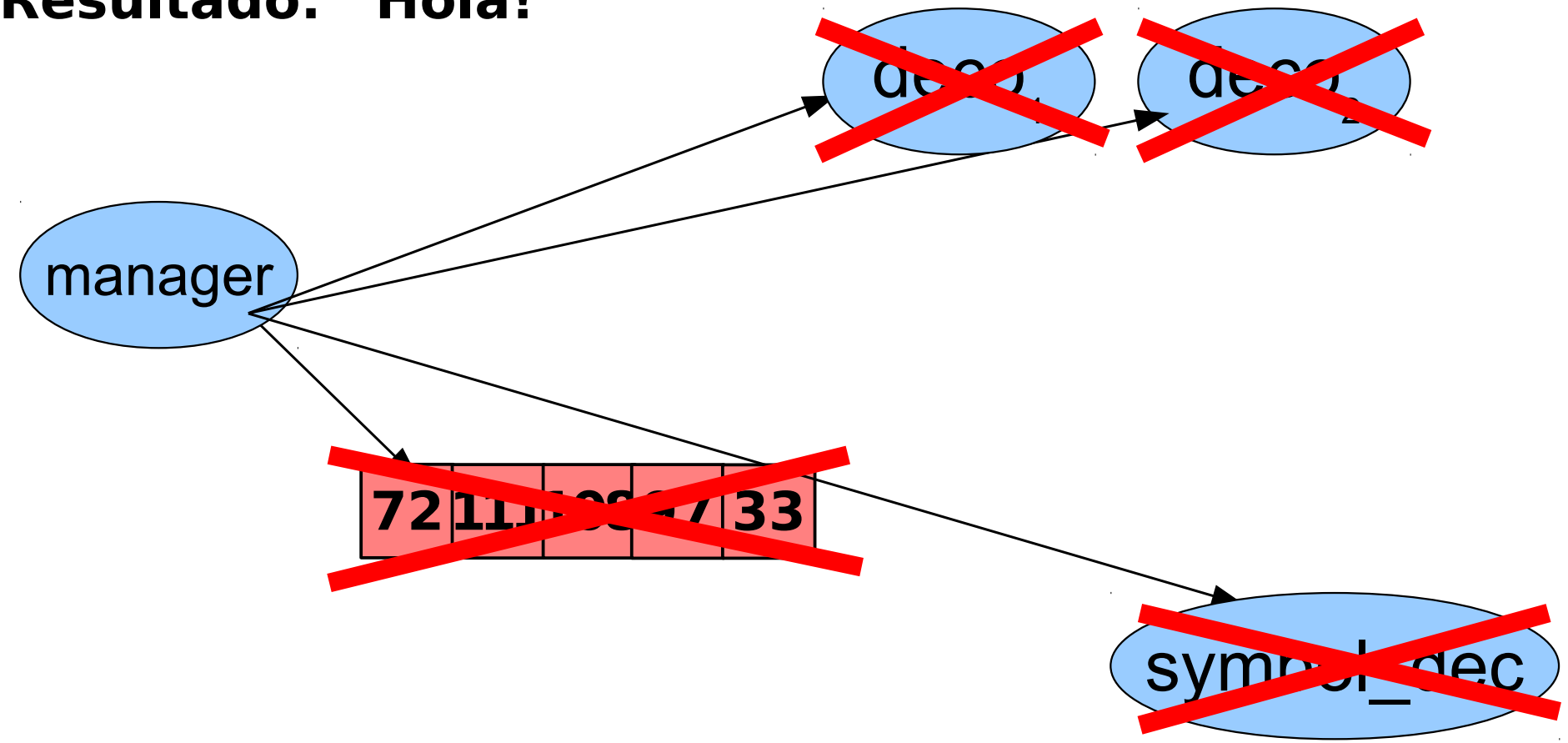
Resultado: “Hola!”



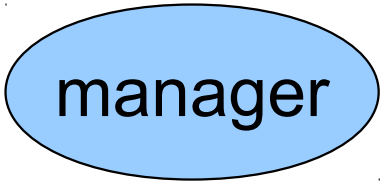
Análisis del Problema de Ejemplo

./exec/manager 34.15.12.1.55 2 2

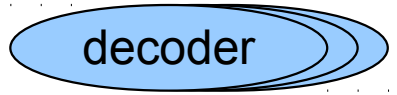
Resultado: “Hola!”



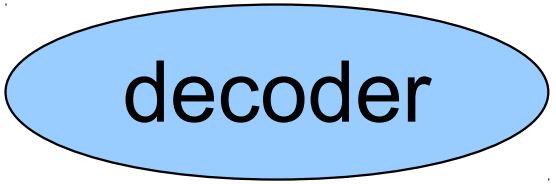
Coordinación de Procesos



1. **Obtener cadena a traducir** (línea órdenes)
2. Creación de **Recursos Compartidos**
   Semáforos
   Segmentos de Memoria Compartida
3. Lanzar **N** procesos **Decoder**
4. Lanzar **1** proceso **SymbolDecoder**
5. **Solicitar traducción** de subvectores.
6. Obtener traducciones y mostrar **resultado**.



Coordinación de Procesos



Bucle

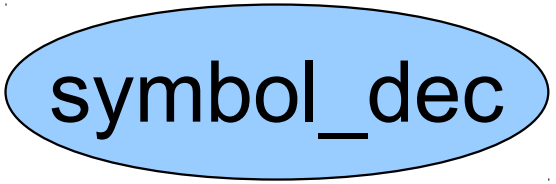
1. Obtener **Orden**
2. Traducir subvector

   Para cada carácter

      Si es de puntuación → Encargar a

      Si no lo es, traducir directamente

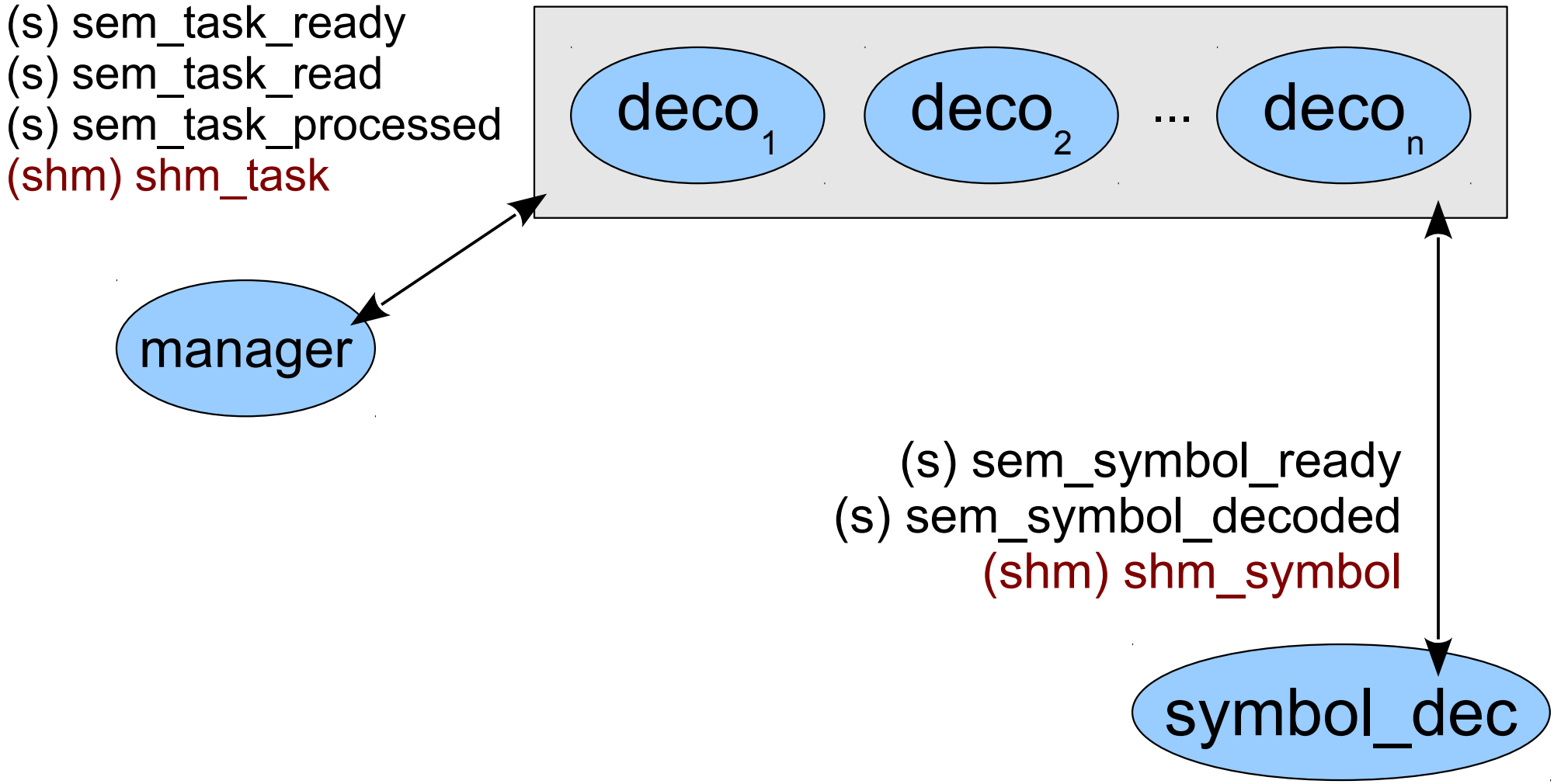
Coordinación de Procesos



Bucle

1. Obtener **Carácter_de_Puntuación**
2. Traducir **Carácter_de_Puntuación**

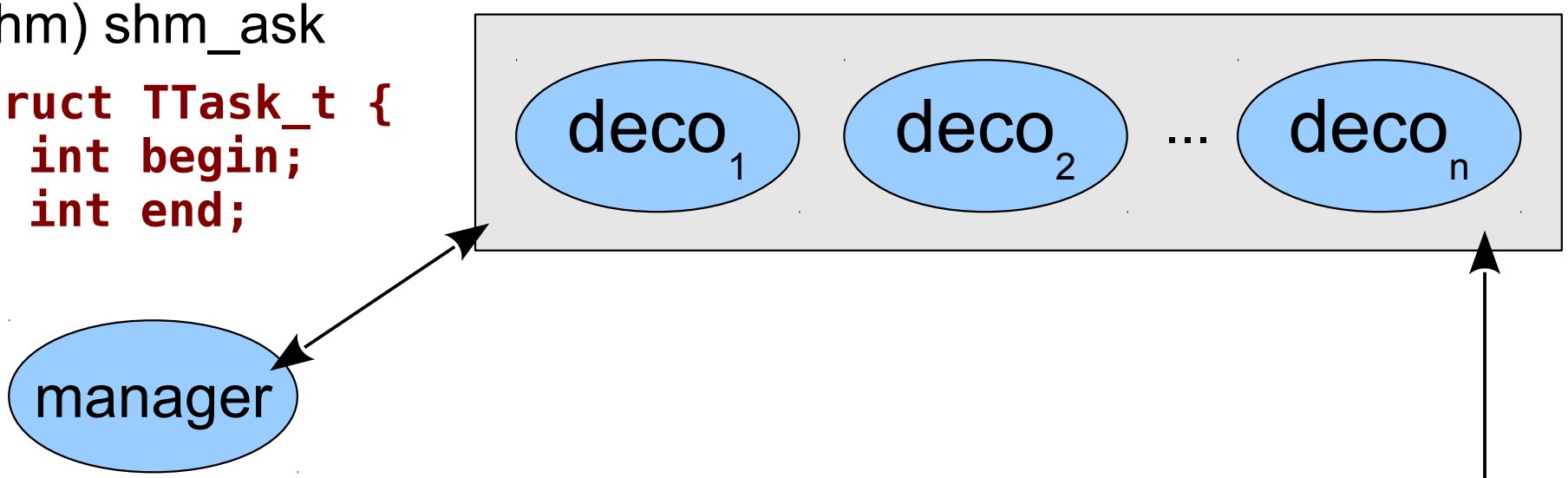
Coordinación de Procesos



Tipos de Datos

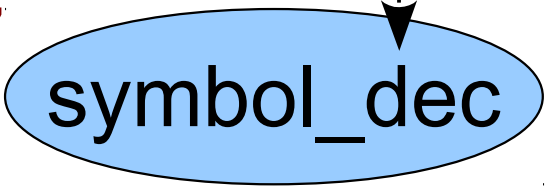
(shm) shm_ask

```
struct TTask_t {
    int begin;
    int end;
}
```

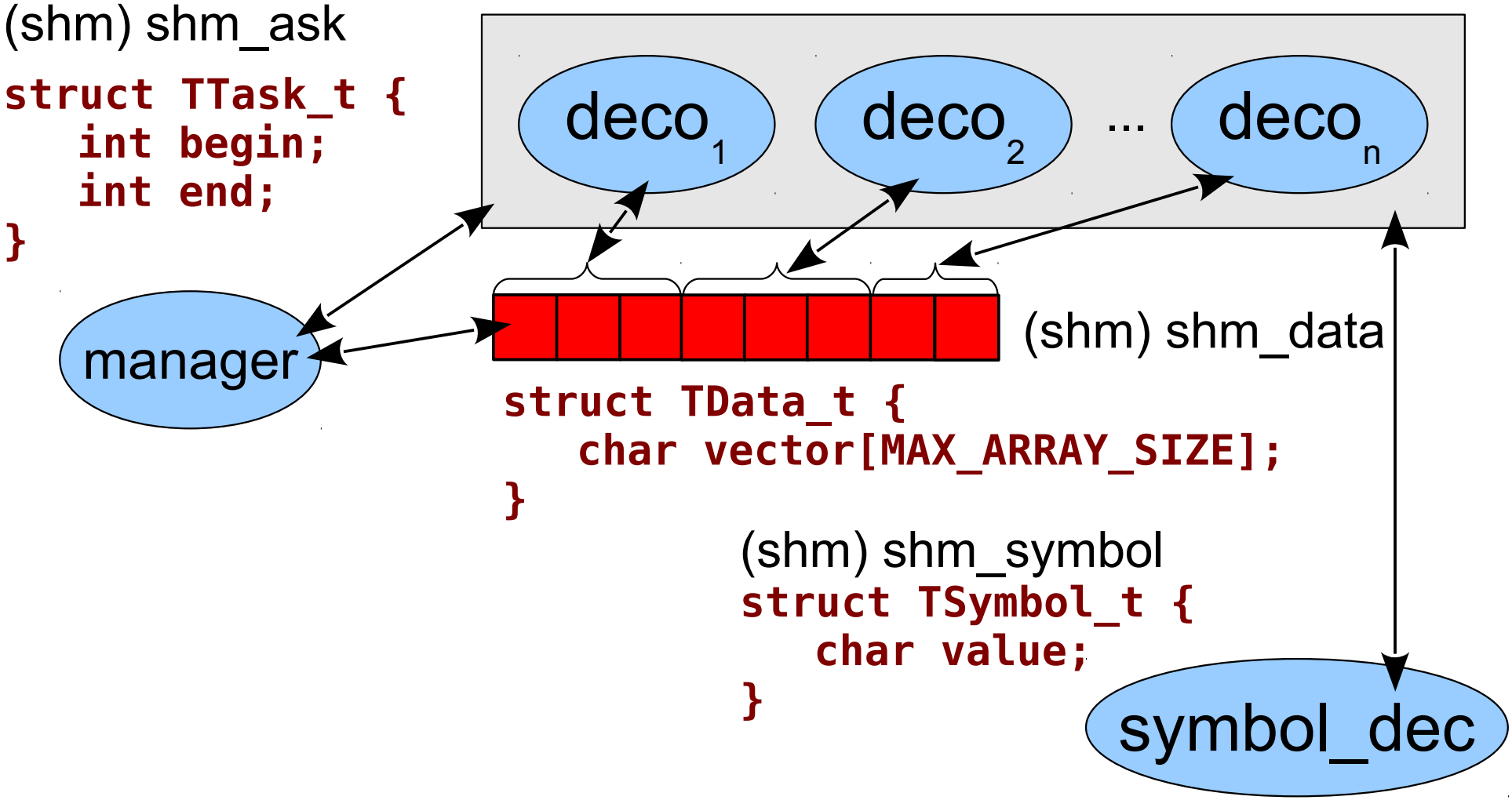


(shm) shm_symbol

```
struct TSymbol_t {
    char value;
}
```



Tipos de Datos



Gestión de Memoria Compartida

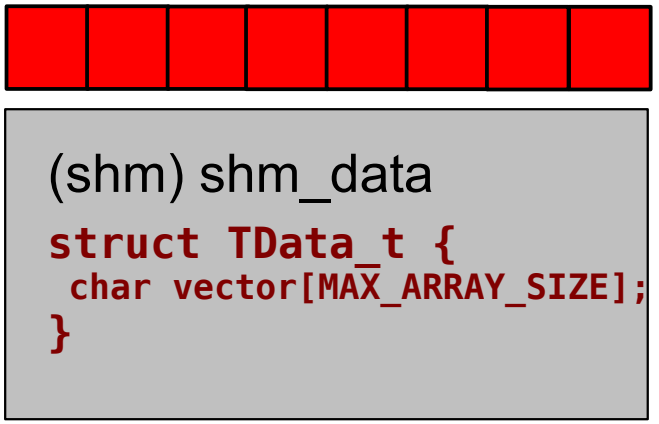
Creación de Segmentos (manager)

```
struct TData_t *data;
int shm_data;

/* Creación del segmento */
shm_data = shm_open(SHM_DATA,
        O_CREAT | O_RDWR, 0644);

/* Tamaño */
ftruncate(shm_data, sizeof(struct TData_t));

/* Mapeo del segmento de memoria compartida */
data = mmap(NULL, sizeof(struct TData_t),
      PROT_READ | PROT_WRITE, MAP_SHARED, shm_data, 0);
```



Eliminación de Segmentos (manager)

```
close(shm_data);   shm_unlink(SHM_DATA);
```

Gestión de Memoria Compartida

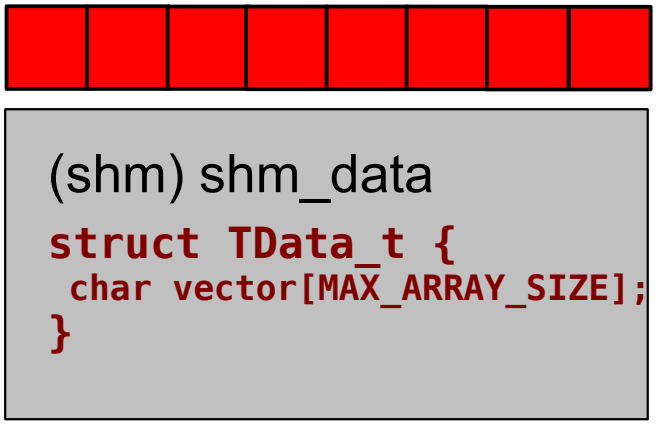
Uso de segmentos creados (ej. traduccion)

```
struct TData_t *data;
int shm_data;

/* Apertura del segmento */
shm_data = shm_open(SHM_DATA,
          O_CREAT|O_RDWR, 0666);

/* Tamaño */
ftruncate(shm_data, sizeof(struct TData_t));

/* Mapeo del segmento de memoria compartida */
data = mmap(NULL, sizeof(struct TData_t),
       PROT_READ | PROT_WRITE, MAP_SHARED, shm_data, 0);
```



Semáforos

manager.c

```
*p_sem_task_ready = create_semaphore(
                        SEM_TASK_READY, 0);
*p_sem_task_read = create_semaphore(
                        SEM_TASK_READ, 0);
*p_sem_task_processed = create_semaphore(
                        SEM_TASK_PROCESSED, 0);
/* The manager process only
initializes the rest,
but it does not use them */
create_semaphore(SEM_MUTEX, 1);
create_semaphore(SEM_SYMBOL_READY, 0);
create_semaphore(SEM_SYMBOL_DECODED, 0);
```

Notificación/recepción de órdenes

manager.c

```
while (current_task < *n_tasks) {
  generate_task(&task);
  current_task++;
  signal_semaphore(sem_task_ready);
  wait_semaphore(sem_task_read);
}
```

Patrón **Rendezvous**

decoder.c

```
wait_semaphore(sem_task_ready);
task_begin = task->begin;
task_end = task->end;
signal_semaphore(sem_task_read);
```

Notificación/recepción de órdenes

¡Independiza la recepción de una tarea de su ejecución!

decoder.c

```
wait_semaphore(sem_task_ready);
task_begin = task->begin;
task_end = task->end;
process_task(task);
signal_semaphore(sem_task_read);
```

Notificación/recepción de órdenes

¡Independiza la recepción de una tarea de su ejecución!

decoder.c

```
wait_semaphore(sem_task_ready);
task_begin = task->begin;
task_end = task->end;
process_task(task);
signal_semaphore(sem_task_read);
```

Notificación de fin de tarea

manager.c

```
/* Esperar a que todos los subvectores */
/* hayan sido calculados. */
int n_tasks_processed = 0;

while (n_tasks_processed < n_tasks) {
    wait_semaphore(sem_task_processed);
    n_tasks_processed++;
}
```

decoder.c

```
signal_semaphore(sem_task_processed);
```

¿Símbolo de puntuación?

decoder.c

```
wait_semaphore(sem_mutex);
        symbol->value = data->vector[i];
        signal_semaphore(sem_symbol_ready);
        wait_semaphore(sem_symbol_decoded);
        data->vector[i] = symbol->value;
signal_semaphore(sem_mutex);
```

Patrón **Rendezvous**

symbol_decoder.c

```
wait_semaphore(sem_symbol_ready);
decode(symbol->value);
signal_semaphore(sem_symbol_decoded);
```