Grupo ARCOS
Universidad Carlos III de Madrid

Lección 3
Ejercicios de paso de mensajes

Sistemas Distribuidos
Grado en Ingeniería Informática

Ejercicio 1

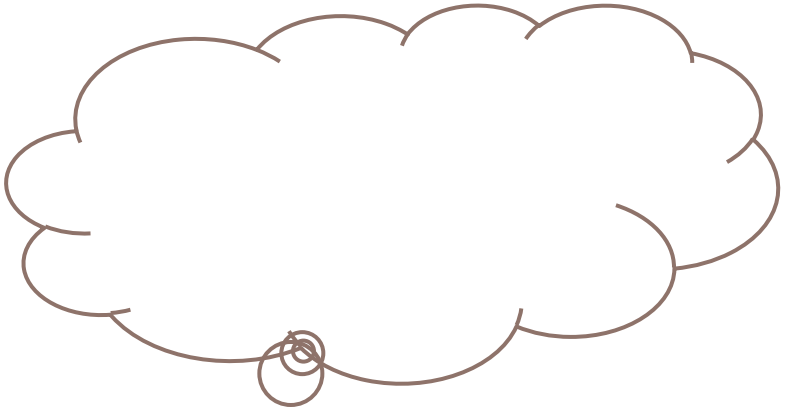
Desarrollar un servidor que permita obtener la hora, la fecha y el día de la semana en la que cae un día determinado.
Diseñar y desarrollar el cliente y el servidor en los dos siguientes supuestos:

a) Se dispone de un sistema con los siguientes servicios:
   - int **Connect** (int pid): este servicio estable una conexión con un proceso con identificador pid. Devuelve un identificador de conexión.
   - int **Accept** (void): este servicio acepta una conexión de un proceso que ejecute el servicio Connect. Devuelve un identificador de conexión.
   - void **Send** (int ic, char *mensaje, int long): este servicio envía un mensaje de una determinada longitud a través del identificador de conexión "ic".
   - void **Receive** (int ic, char *mensaje, int long): este servicio recibe un mensaje de una determinada longitud de la conexión con identificador "ic".

   Asuma que los identificadores de los procesos son números enteros y que el servidor viene identificado por el número 1000

b) Considere un sistema que utiliza colas de mensajes POSIX.

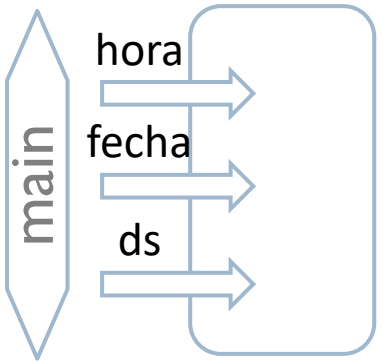
Diseño progresivo



1. Sistema
2. Distribuido
3. Con colas de mensajes POSIX

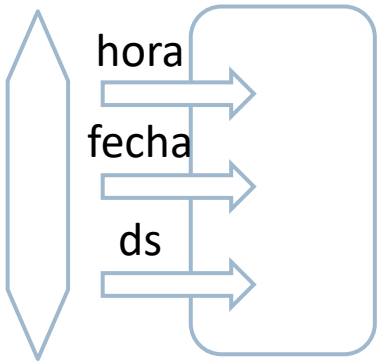
Diseño NO distribuido (v0.2)

Biblioteca usada desde programa



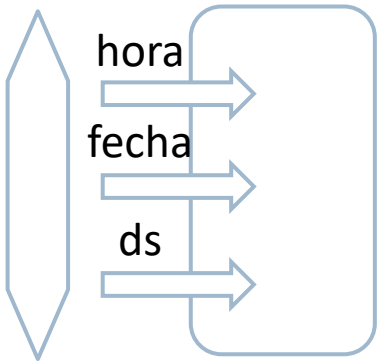
1. **Sistema**
2. Distribuido
3. Con colas de mensajes POSIX

Diseño no distribuido…




```
#include <time.h>

// Devuelve hora.
int hora ( char *hour )
{
    time_t now = time(NULL);
    struct tm *tm_struct = localtime(&now);
    sprint(hour,"%d", tm_struct->tm_hour);
    return 1;
}

// Devuelve la fecha.
int fecha ( char *fecha )
{
    time_t  clk = time(NULL);
    strcpy(fecha, ctime(clk)) ;
    return 1;
}
```

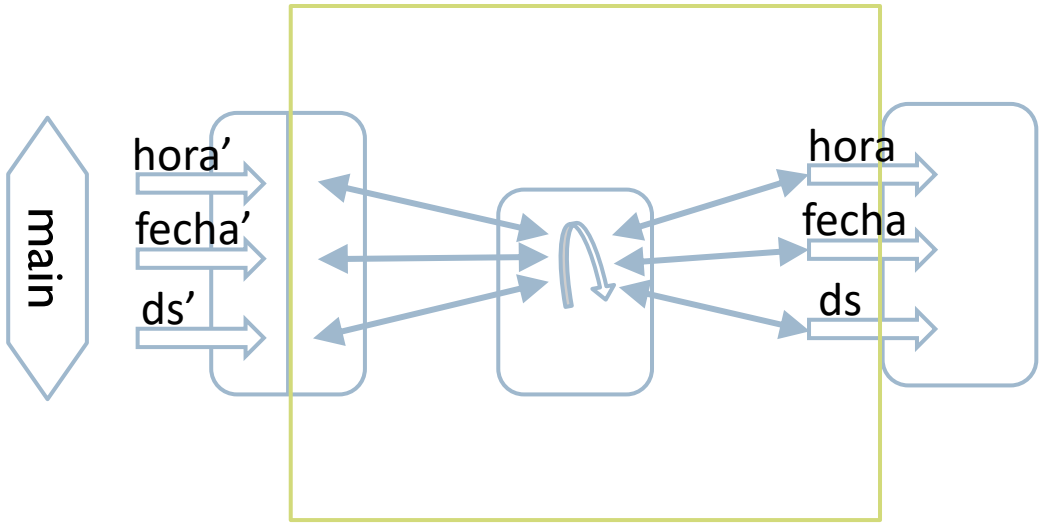
Diseño no distribuido…




```
#include <time.h>

// Devuelve hora.
int ds ( char *dia_semana )
{
    // https://stackoverflow.com/questions/6054016/c-program-to-find-day-of-week-given-date
    struct tm tm;
    memset((void *) &tm, 0, sizeof(tm));
    if (strptime(str, "%d-%m-%Y", &tm) != NULL)
    {
        time_t t = mktime(&tm);
        if (t >= 0) {
            int ds = localtime(&t)->tm_wday; // Sunday=0, Monday=1, etc.
            sprint(dia_semana, "%d", ds);
            return 1;
        }
    }

    return -1;
}
```

Diseño distribuido (v0.5)

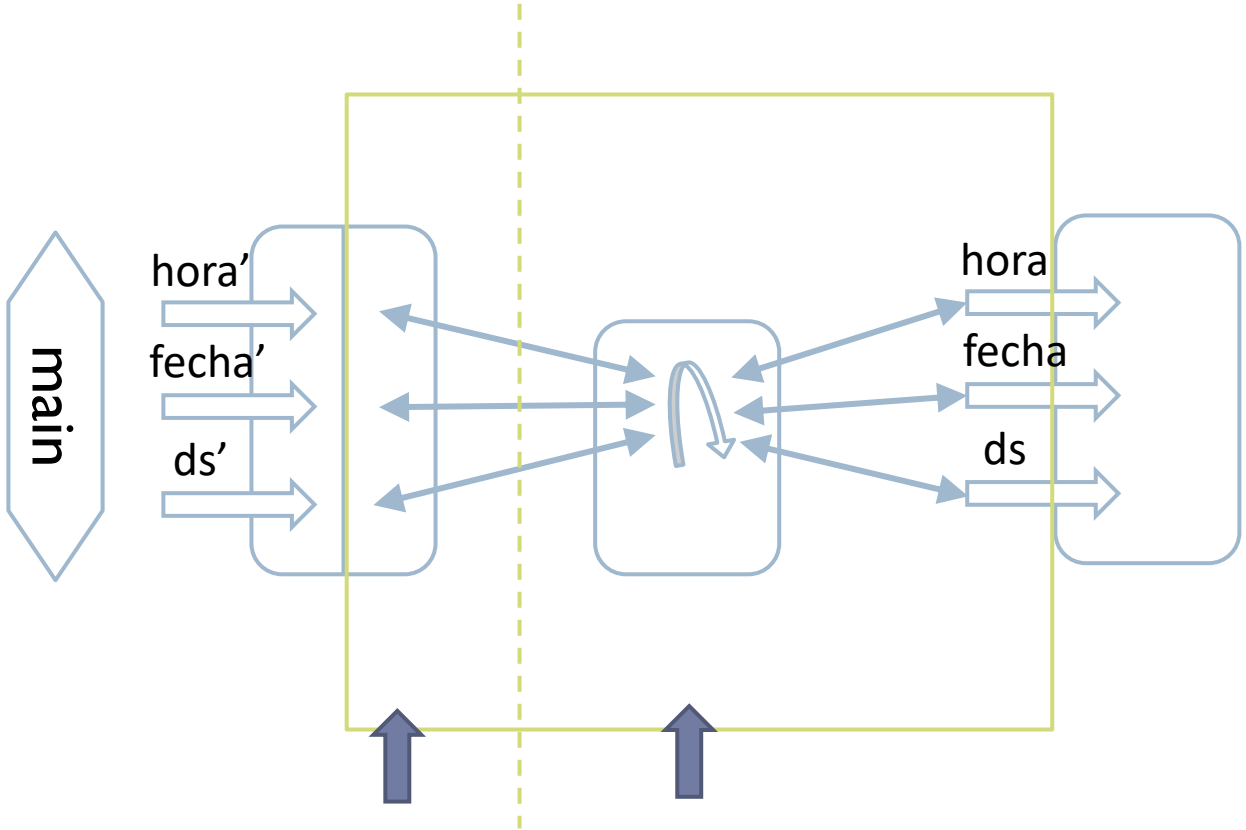
Usar un proxy para los servicios remotos



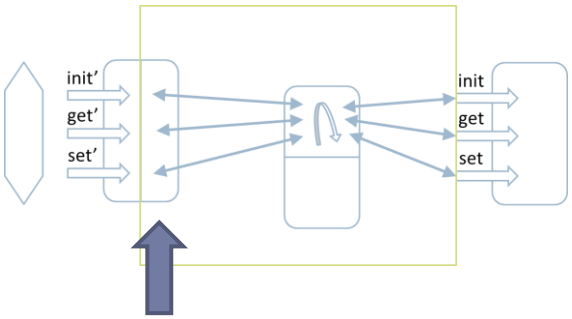
1. Sistema
2. **Distribuido**
3. Con colas de mensajes POSIX

Diseño distribuido (v0.5)

Usar un proxy para los servicios remotos



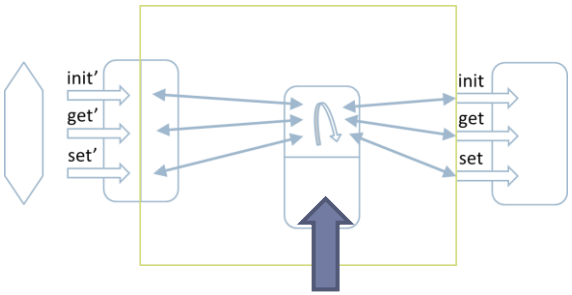
Diseño distribuido…




```
// msg[0]          -> status (0...255)
// msg[1]…[31] -> value


int send_recv ( char *op,
                char *value )
{
    char msg[1024]
    strcpy(msg, op)

    c1 = Connect(1000)
    Send(c1, msg, 1024)
    Receive(c1, msg, 1024)

    strcpy(value, msg+1)
    return (int)msg[0]
}
```

```
int hora ( char *hour )
{
    return send_recv("hora", hour)
}

int fecha ( char *fecha )
{
    return send_recv( "fecha", fecha)
}

int ds ( char *dia_semana )
{
    return send_recv("ds", día_semana)
}
```

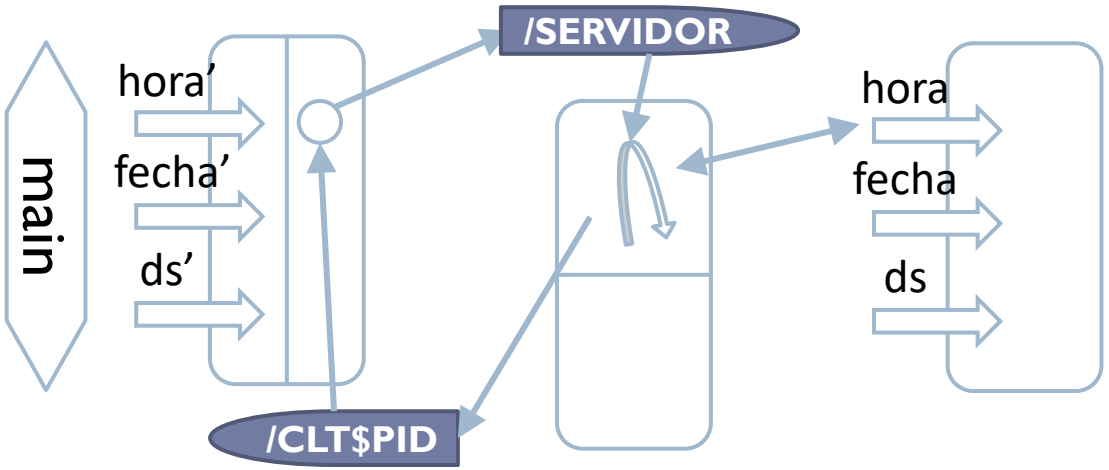
Diseño distribuido…




```
int main ( int argc, char *argv )
{
    while (TRUE)
    {
        c1 = Accept()
        Receive(c1, msg, 1024)
        switch(msg)
        {
            case "hora":  msg[0] = (char) hora(msg+1) ;
                          break;
            case "fecha": msg[0] = (char) fecha(msg+1) ;
                          break;
            case "ds":    msg[0] = (char) ds(msg+1) ;
                          break;
        }

        Send(c1, msg, 1024) ;
    }
}
```

Diseño distribuido (v0.8)

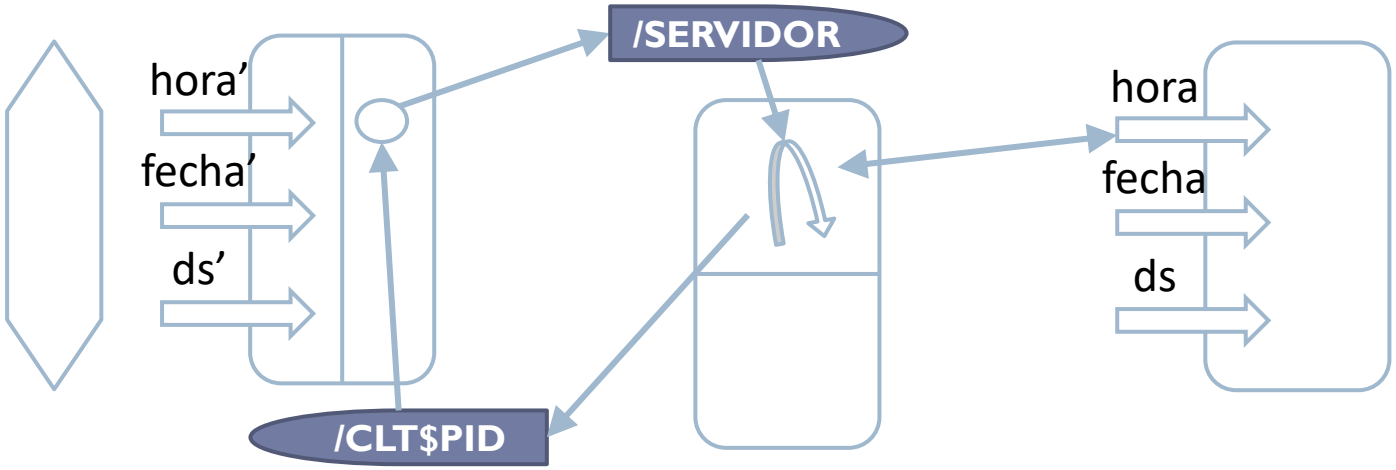
Adaptar proxy a colas de mensajes POSIX



1. Sistema
2. Distribuido
3. **Con colas de mensajes POSIX**

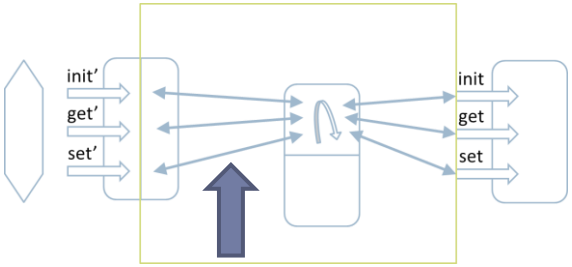
Diseño distribuido (v0.8)

Adaptar proxy a colas de mensajes POSIX



- El mensaje de petición ha de valer para todos los servicios
  - Establecer un identificador numérico para cada servicio
  - Establecer los parámetros para cada servicio y generar una petición la fusión de todos + identificador de servicio.
  - Establecer las respuestas para cada servicio y generar una respuesta fusión.

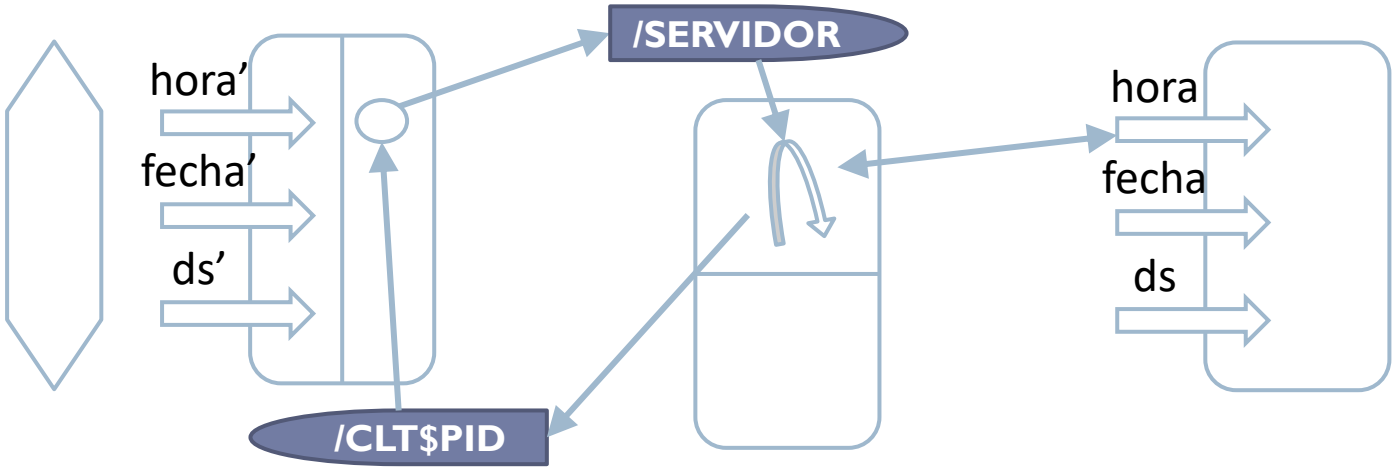
Diseño distribuido…




```
//  petición = op + q_name
struct peticion
{
      int     op;
      char  q_name[MAX];
} ;


// respuesta = (value, status)
struct respuesta
{
      char  value[32];
      char  status;
} ;
```

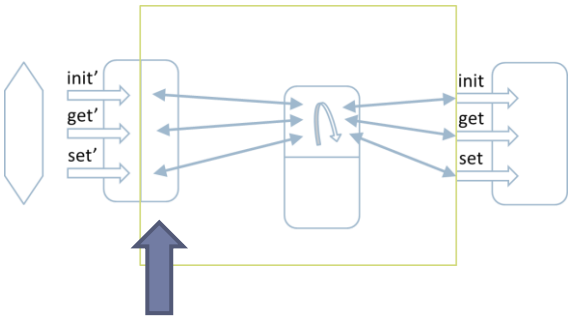
Diseño distribuido (v0.8)

Adaptar proxy a colas de mensajes POSIX



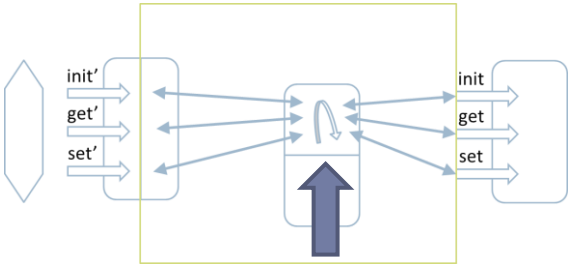
- Las colas POSIX son unidireccionales
  - Una cola general de peticiones creada por el servidor
  - Por cada cliente activo una cola (efímera) para recibir la respuesta.
    - La cola es privada para cada el cliente con nombre único (usar getpid())

Diseño distribuido…



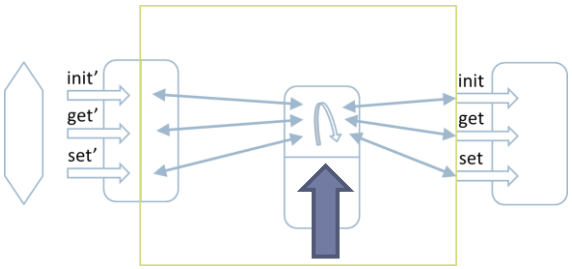

```
int hora  ( char *hour )
{
    struct petición p;
    struct respuesta r;
    char qr_name[1024];  int prio;

    sprintf(qr_name, "%s%d", "/CLIENTE_", getpid()) ;
    int qs = mq_open("/SERVIDOR", O_CREAT|O_WRONLY, 0700, NULL) ;
    if (qs == -1) { return -1 ; }
    int qr = mq_open(qr_name, O_CREAT|O_RDONLY) ;
    if (qr == -1) { mq_close(qs) ; return -1; }
    p.op = 1;
    strcpy(p.q_name, qr_name);
    mq_send    (qs,  (char *)&p,  sizeof(struct petición),    0) ;
    mq_receive(qr,   (char *)&r,  sizeof(struct respuesta),  &prio) ;
    mq_close(qs);
    mq_close(qr);
    mq_unlink(qr_name);
    strcpy(hour, r.value) ;
    return r.status ;
}
```

Diseño distribuido…




```
int main ( int argc, char *argv[] )
{
    struct petición p;
    int prio;

    int qs = mq_open("/SERVIDOR", O_CREAT | O_RDONLY, 0700, NULL) ;
    if (qs == -1) { return -1 ; }
    while (1)
    {
        mq_receive(qs, &p, sizeof(p), &prio) ;
        tratar_petición(&p) ;
    }
}
```

Diseño distribuido…




```
void tratar_petición ( struct petición * p )
{
    struct respuesta r ;
    unsigned int prio = 0 ;
    switch (p->op)
    {
        case 1: // HORA
                r.status = real_hora(p->value) ;
                break ;
        case 2: // FECHA
                r.status = real_fecha(p.value) ;
                break ;
        case 3: // DIA_SEMANA
                r.status = real_ds(p->value) ;
                break ;
    }

    int qr = mq_open(p->q_name, O_CREAT|O_WRONLY, 0700, NULL) ;
    mq_send(qr, &r, sizeof(struct respuesta), prio) ;
    mq_close(qr);
}
```

Grupo ARCOS
Universidad Carlos III de Madrid

Lección 3
Ejercicios de paso de mensajes

Sistemas Distribuidos
Grado en Ingeniería Informática