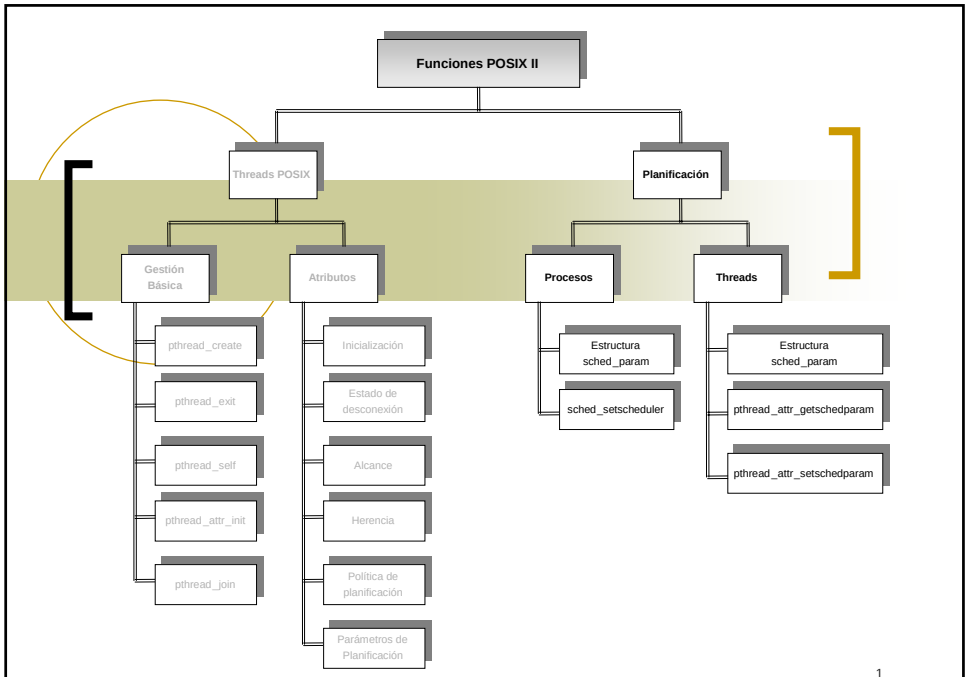

Planificación en UNIX/POSIX:

Procesos

Conceptos de planificación en Unix (I)

- Unix utiliza un esquema de planificación por clases de prioridades (n colas con prioridad asociada donde cada cola se planifica de forma independiente)
- Cada proceso tiene una política y unos atributos de planificación asociados los cuales se pueden modificar de forma independiente
- POSIX define tres políticas
  - **SCHED_FIFO**: FIFO expulsivo con prioridades
  - **SCHED_RR**: Round Robin con prioridades (no se permite cambiar el quantum)
  - **SCHED_OTHER**: tiempo compartido sin prioridades (0) (por defecto en UNIX)

Planificación de Procesos (I)

- Los parámetros de planificación están encapsulados en una estructura

```
struct sched_param {
            int sched_priority;
      }
```

- La llamada que permite cambiar la prioridad y la política de planificación de un Proceso es sched_setscheduler

```
int sched_setscheduler (pid_t pid, int policy,
                        sched_param *param);
int sched_getscheduler (pid_t pid);
```

  - Si pid es 0 cambia la política de planificación del proceso que ejecuta la función
  - Devuelve 0 si el cambio ha tenido éxito, -1 en caso de error

Planificación de Procesos (II)

- Ejemplo: Planificar un proceso con política SCHED_FIFO y prioridad 17

```
#include <sched.h>
struct sched_param scheduling_parameters;
int res;

scheduling_parameters.sched_priority = 17;

res = sched_setscheduler( getpid(), SCHED_FIFO,
                          &scheduling_parameters);
```


Planificación de Procesos (III)

- Si un proceso se ejecuta con política *SCHED_FIFO* este se ejecuta hasta que se pasa a modo Suspendido por una espera de E/S, u otro proceso de más **prioridad** pasa a modo de Espera
- Para liberar el uso de la CPU en un punto intermedio de la ejecución se puede usar la llamada del sistema ***sched_yield()***.
  - El proceso se coloca al final de la lista FIFO

```
int sched_yield (void);
```

  - Devuelve 0 si el cambio ha tenido éxito, -1 en caso de error

Planificación de Procesos (IV)

- Si un proceso se ejecuta con política *SCHED_RR* este se ejecuta hasta que se cumple su *quantum*, pasa a modo Suspendido por una espera de E/S, u otro proceso de más **prioridad** pasa a modo de Espera
  - Para liberar el uso de la CPU en un punto intermedio de la ejecución se puede usar la llamada del sistema ***sched_yield()***.
- Se puede obtener el valor del quantum. POSIX no especifica un método para su modificación.

```
int sched_rr_get_interval (pid_t pid, struct timespec *tp);

struct timespec {
        time_t tv_sec; /* seconds */
        long tv_nsec; /* nanoseconds */
};
```

```
struct timespec quantum;
if(sched_rr_get_interval(getpid(), &quantum)==0)
        printf("Quantum: %f ms\n",(float)quantum.tv_nsec/1000000L);
```


Ejemplo:

Ejemplo planificación de procesos con prioridades FIFO

:compilar con librería: **–lposix4**
:compilar en linux con librería: **–lrt**

```c
#include <sys/types.h>
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <sched.h>

int  main (void)
{
  pid_t pid;
  struct sched_param scheduling_parameters;
  int i,res;
  scheduling_parameters.sched_priority = 17;
  res = sched_setscheduler( getpid(), SCHED_FIFO,
                                    &scheduling_parameters);
  pid = fork();
  switch (pid)
  {
        case -1: perror ("No se ha podido crear el hijo");
                 break;
        case 0:  printf("Soy el hijo, mi PID es %d y mi PPID es %d\n",
                                    getpid(), getppid()); sleep(1);
                 for(i=0;i<10;i+=2)
                 { printf("%ld\n",i); fflush(stdout); sleep(1); }
                 break;

        default: printf ("Soy el padre, mi PID es %d y el PID de mi
                          hijo es %d\n", getpid(), pid); sleep(1);
                 for(i=1;i<10;i+=2)
                 { printf("%ld\n",i); fflush(stdout); sleep(1); }
                 wait(0);  // espera que termine el hijo
  }
  exit(0);
}
```


Ejemplo planificación procesos: Resultado

```
[/u0/sitr/sitr001/procesos]planificador
Soy el hijo, mi PID es 11206 y mi PPID es 11205
Soy el padre, mi PID es 11205 y el PID de mi hijo es 11206
1
0
2
3
5
4
6
7
9
8
[/u0/sitr/sitr001/procesos]
```

```c
#include <sys/types.h>
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <sched.h>

int  main (void)
{
  pid_t pid;
  struct sched_param scheduling_parameters;
  int i,res;
  scheduling_parameters.sched_priority = 17;
  res = sched_setscheduler( getpid(), SCHED_FIFO,
                            &scheduling_parameters);
  pid = fork();
  switch (pid)
  {
        case -1: perror ("No se ha podido crear el hijo");
                 break;
        case 0:  printf("Soy el hijo, mi PID es %d y mi PPID es %d\n",
                                 getpid(), getppid());
                 for(i=0;i<10;i+=2)
                 { printf("%ld\n",i); fflush(stdout); /*sleep(1);*/ }
                 break;

        default:  printf ("Soy el padre, mi PID es %d y el PID de mi
                           hijo es %d\n", getpid(), pid);
        for(i=1;i<10;i+=2)
                 { printf("%ld\n",i); fflush(stdout); /*sleep(1);*/ }
                 wait(0);  // espera que termine el hijo
  }
  exit(0);
}
```


Ejemplo planificación procesos: Resultado

```
sitr001@lorca:~$ ./planificador
Soy el hijo, mi PID es 15341 y mi PPID es 15340
0
2
4
6
8
Soy el padre, mi PID es 15340 y el PID de mi hijo es 15341
1
3
5
7
9
sitr001@lorca:~$
```

Planificación en UNIX/POSIX:

Threads

Atributos de los Pthreads

- Propiedades asociadas a los threads:

Propiedad	Función
Inicialización	pthread_attr_init pthread_attr_destroy
Tamaño de la pila	pthread_attr_setstacksize pthread_attr_getstacksize
Dirección de la pila	pthread_attr_setstackaddr pthread_attr_getstackaddr
Estado de desconexión	pthread_attr_setdetachstate pthread_attr_getdetachstate
Alcance	pthread_attr_setscope pthread_attr_getscope
Herencia	pthread_attr_setinheritsched pthread_attr_getinheritsched
Política de planificación	pthread_attr_setschedpolicy pthread_attr_getschedpolicy
Parámetros de planificación	pthread_attr_setschedparam pthread_attr_getschedparam

Planificación de threads (I): Alcance

- Permite trabajar con distintos modelos de threads (si el sistema lo permite)
  - El atributo *contentionscope* puede ser
    - PTHREAD_SCOPE_PROCESS: thread a nivel de usuario
    - PTHREAD_SCOPE_SYSTEM: thread a nivel de núcleo
  - pthread_attr_getscope: consulta el alcance de contención
  - pthread_attr_setscope: establece el alcance de contención

```
int pthread_attr_setscope (pthread_attr_t *attr,
                           int contentionscope);
int pthread_attr_getscope (const pthread_attr_t *attr,
                           int *contentionscope);
```


Planificación de threads (II)

- La política y atributos de planificación de un thread se encapsulan en un objeto (estructura) de tipo *struct sched_param* que contiene
  - *sched_policy*: SCHED_FIFO, SCHED_RR, SCHED_OTHER
  - *sched_priority*: número entero
- El comportamiento real de la política de planificación depende del alcance del thread
- Los parámetros de planificación pueden heredarse:

```
int pthread_attr_setinheritsched (pthread_attr_t *attr,
                                  int inheritsched);
int pthread_attr_getinheritsched (pthread_attr_t *attr,
                                  int *inheritsched);
```

  - ***inheritsched*** -> valores:
    - PTHREAD_INHERIT_SCHED
    - PTHREAD_EXPLICIT_SCHED

Planificación de threads (III)

- Cambio de política de planificación de un thread:

```
int pthread_attr_setschedpolicy (pthread_attr_t *attr,
                    int policy);
int pthread_attr_getschedpolicy (pthread_attr_t *attr,
                    int *policy);
```

- La función para modificar los parámetros adicionales de planificación de un thread (*p.e. prioridad*) es:

```
int pthread_attr_setschedparam (pthread_attr_t *attr,
                    const struct sched_param *param);
int pthread_attr_getschedparam (pthread_attr_t *attr,
                            struct sched_param *param);
```


EJEMPLO:

Planificación threads:
cambio política y prioridad

:compilar con librerías: **–lposix4 -lpthread**
:compilar en linux con librerías: **–lrt -lpthread**

```c
#include <pthread.h>
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include <sys/types.h>
#include <sched.h>
#define HIGHPRIORITY 10

pthread_attr attr;
pthread_t thread;
struct sched_param param;

void *func (void*); // Prototipo ejemplo de func

int main (void)
{
  if (pthread_attr_init(&attr))
        {perror("No puedo inicializar atributos"); exit(-1);}
  if (pthread_create(&thread, &attr, func, NULL))
        { perror("No puedo crear el thread"); exit(-1);}

  if(pthread_attr_setschedpolicy (&attr, SCHED_FIFO))
                perror("No puedo cambiar el planificador");
  if (pthread_attr_getschedparam(&attr, &param))
                perror ("No puedo obtener los parámetros");
  else
  {
        param.sched_priority = HIGHPRIORITY;
        if (pthread_attr_setschedparam(&attr, &param))
                perror("No puedo cambiar la prioridad");
  }
}
```