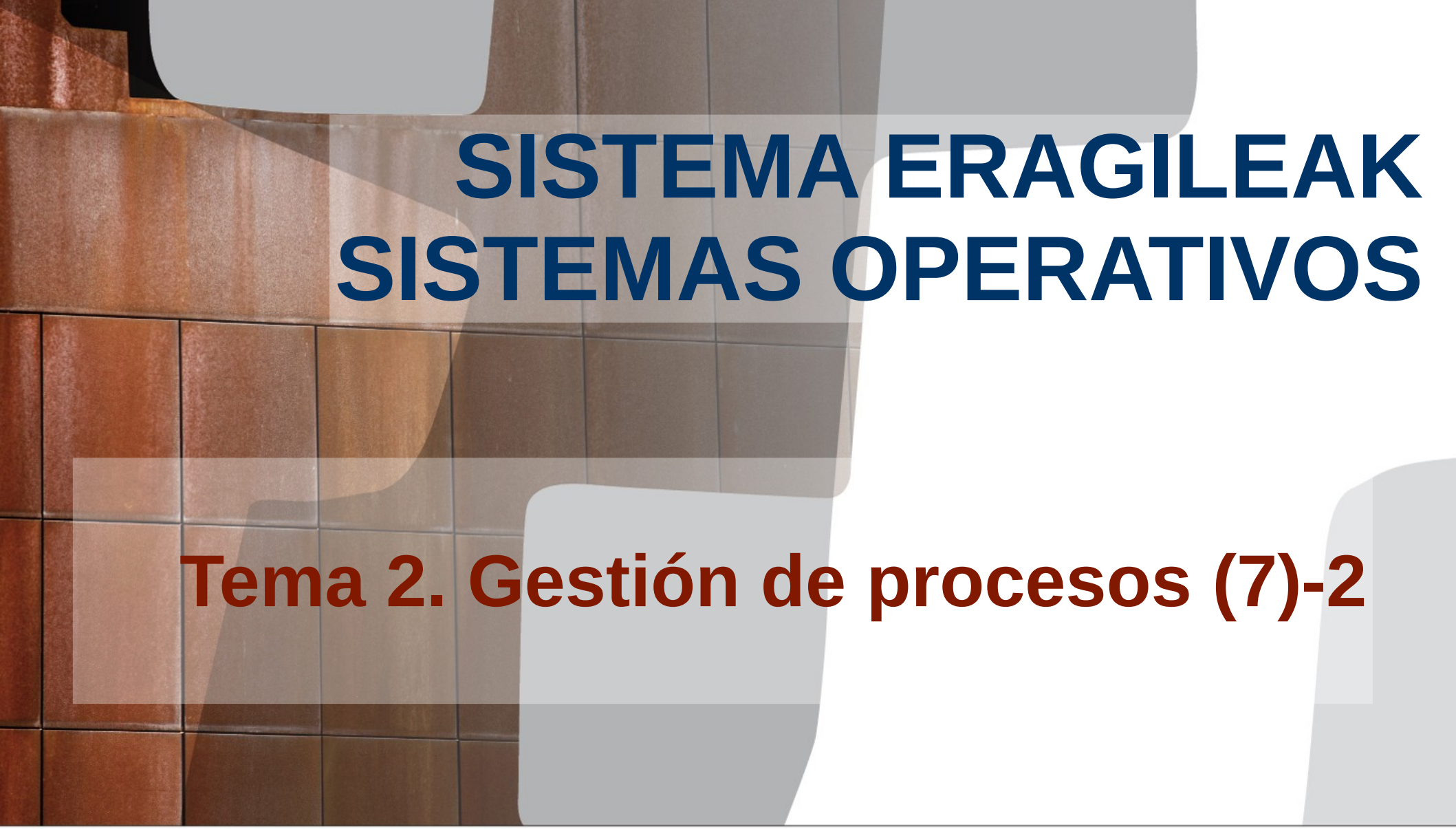


SISTEMA ERAGILEAK
SISTEMAS OPERATIVOS

Tema 2. Gestión de procesos (7)-2

Informatika Ingeniaritzako Gradua
Grado en Ingeniería en Informática
Informatika Fakultatea

eman ta zabal zazu
Universidad del País Vasco
Euskal Herriko Unibertsitatea

NAZIOARTEKO BIKAINTASUN CAMPUSA
CAMPUS DE EXCELENCIA INTERNACIONAL

Variables de condición

```
#include <pthread.h>
```

Declaración de una variable de condición

```
pthread_cond_t cond;
```

- **`cond`**: Puntero a una estructura de tipo `pthread_cond_t`.
- Una variable de condición debe estar asociada con un **`mutex`** para evitar condiciones de carrera.

Crear una variable de condición

```
int pthread_cond_init(pthread_cond_t *cond,
      pthread_condattr_t *attr);
```

- **`cond`**: Puntero a un objeto variable de condición.
- **`attr`**: Puntero a los atributos de la variable de condición. Si es NULL, se utilizan los atributos predeterminados.

- Devuelve `0` en caso de éxito.

Esperar a una variable de condición

```
int pthread_cond_wait(pthread_cond_t *cond,
      pthread_mutex_t *mutex);
```

- De manera atómica, desbloquea el **`mutex`** (similar a **`pthread_unlock_mutex`**) y espera a que se señale a la variable de condición **`cond`**.
- El hilo se bloquea y no consume tiempo de CPU hasta que se señale **`cond`**.
- El **`mutex`** debe estar bloqueado por el hilo llamante en el momento de la llamada.
- Antes de volver al hilo llamante, **`pthread_cond_wait`** vuelve a adquirir el **`mutex`** (similar a **`pthread_lock_mutex`**).

- **`cond`**: Puntero a un objeto variable de condición.
- **`mutex`**: Puntero a un objeto mutex.

Señalar a una variable de condición

```
int pthread_cond_signal(pthread_cond_t *cond);
```

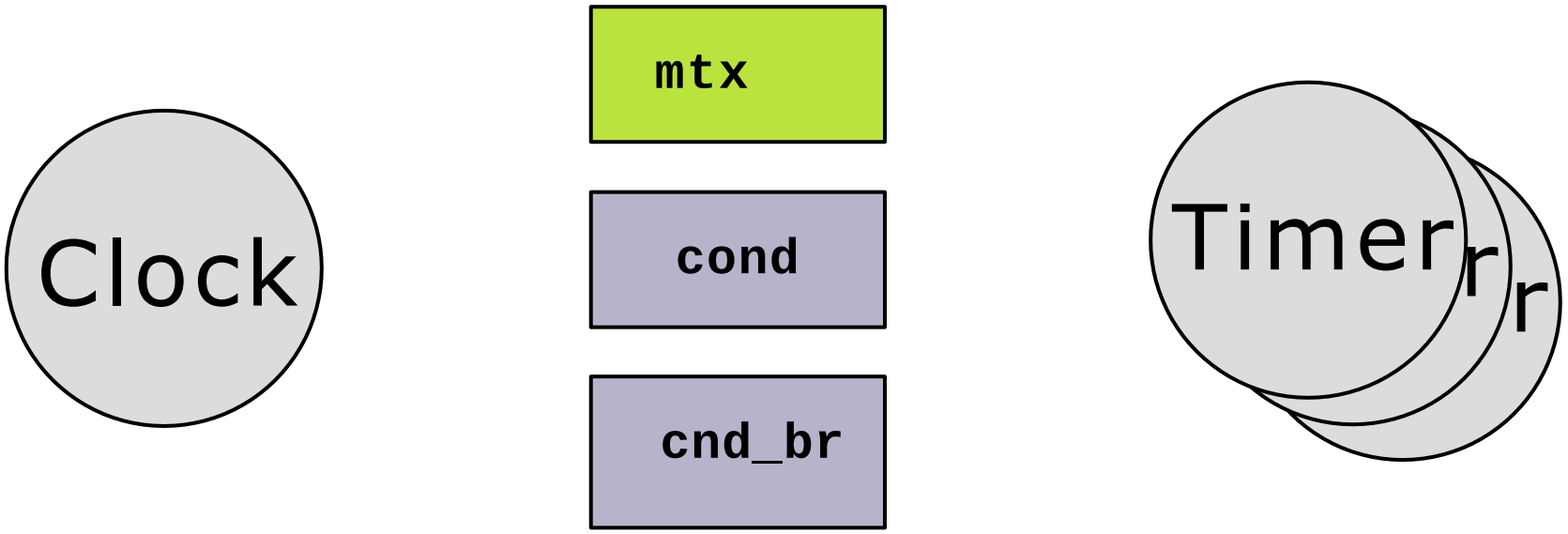
- Reinicia uno de los hilos que están esperando en la variable de condición **cond**.
  - Si no hay hilos esperando en **cond**, no sucede nada.
  - Si hay varios hilos esperando en **cond**, se reinicia uno cualquiera sin especificar cuál.
- **cond**: Puntero a un objeto variable de condición.

Señalización general

```
int pthread_cond_broadcast(pthread_cond_t *cond);
```

- Reinicia todos los hilos bloqueados que están esperando en la variable de condición **cond**.
  - Si no hay hilos esperando en **cond**, no sucede nada.
- **cond**: Puntero a un objeto variable de condición.

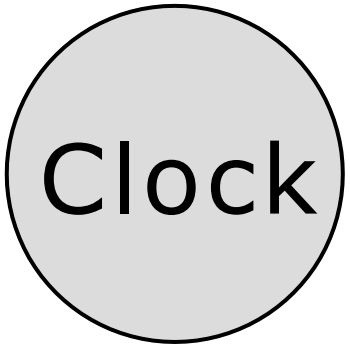
Conditional mutex



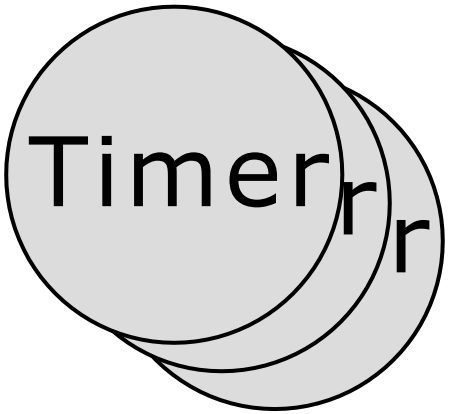

```
while(1){
  mutex_lock(&mtx);
  while(done < timers_num)
    cond_wait(&cond, &mtx);
  ...
  done = 0;
  cond_broadcast(&cnd_br);
  mutex_unlock(&mtx);
}
```

```
mutex_lock(&mtx);
while(1){
  done++;
  ...
  cond_signal(&cond);
  cond_wait(&cnd_br, &mtx);
}
```

Conditional mutex



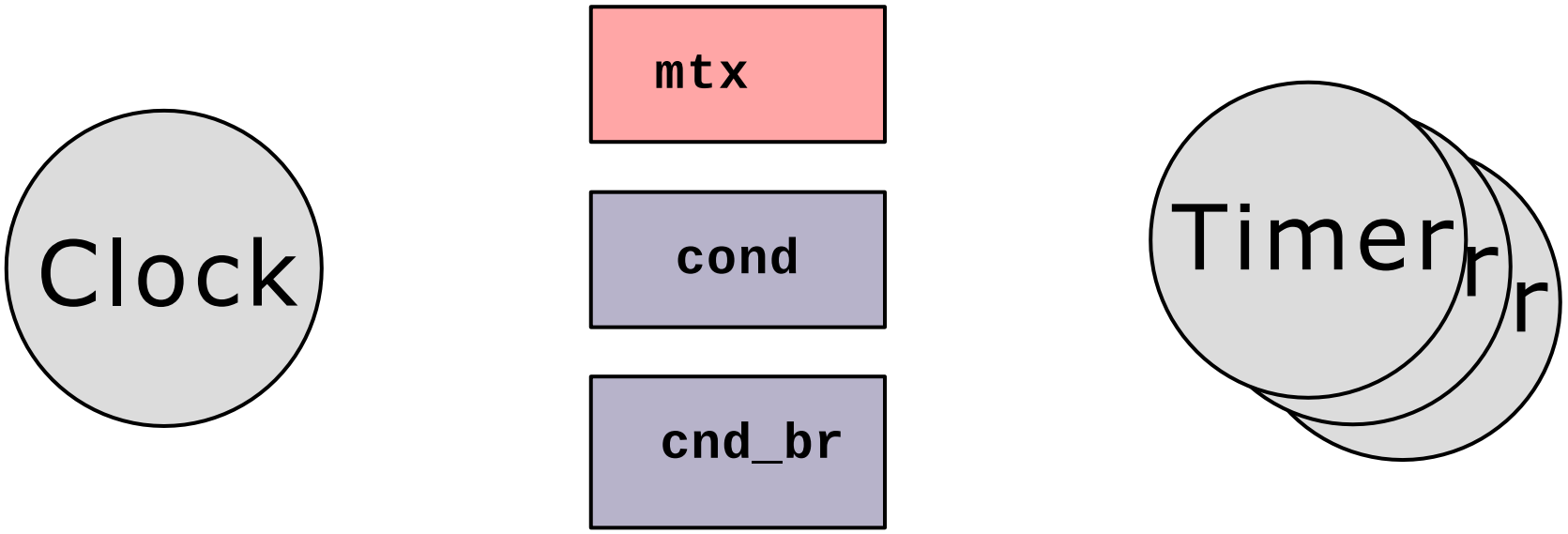

```
init(mtx);
init(cond);
init(cnd_br);

done  = 0;

timers_num = MAX_TIMERS;
```




```
while(1){
  mutex_lock(&mtx);
  while(done < timers_num)
    cond_wait(&cond, &mtx);
  ...
  done = 0;
  cond_broadcast(&cnd_br);
  mutex_unlock(&mtx);
}
```

```
mutex_lock(&mtx);
while(1){
  done++;
  ...
  cond_signal(&cond);
  cond_wait(&cnd_br, &mtx);
}
```

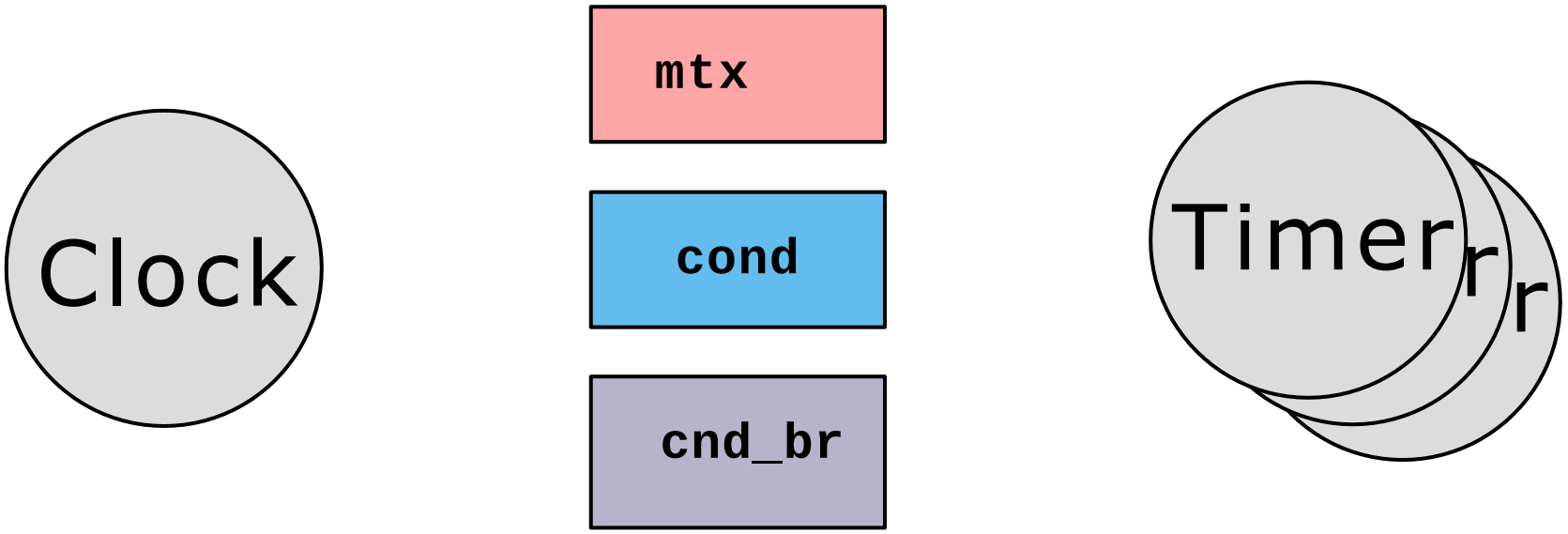
Conditional mutex




```
while(1){
  mutex_lock(&mtx);
  while(done < timers_num)
    cond_wait(&cond, &mtx);
  ...
  done = 0;
  cond_broadcast(&cnd_br);
  mutex_unlock(&mtx);
}
```

```
mutex_lock(&mtx);
while(1){
  done++;
  ...
  cond_signal(&cond);
  cond_wait(&cnd_br, &mtx);
}
```

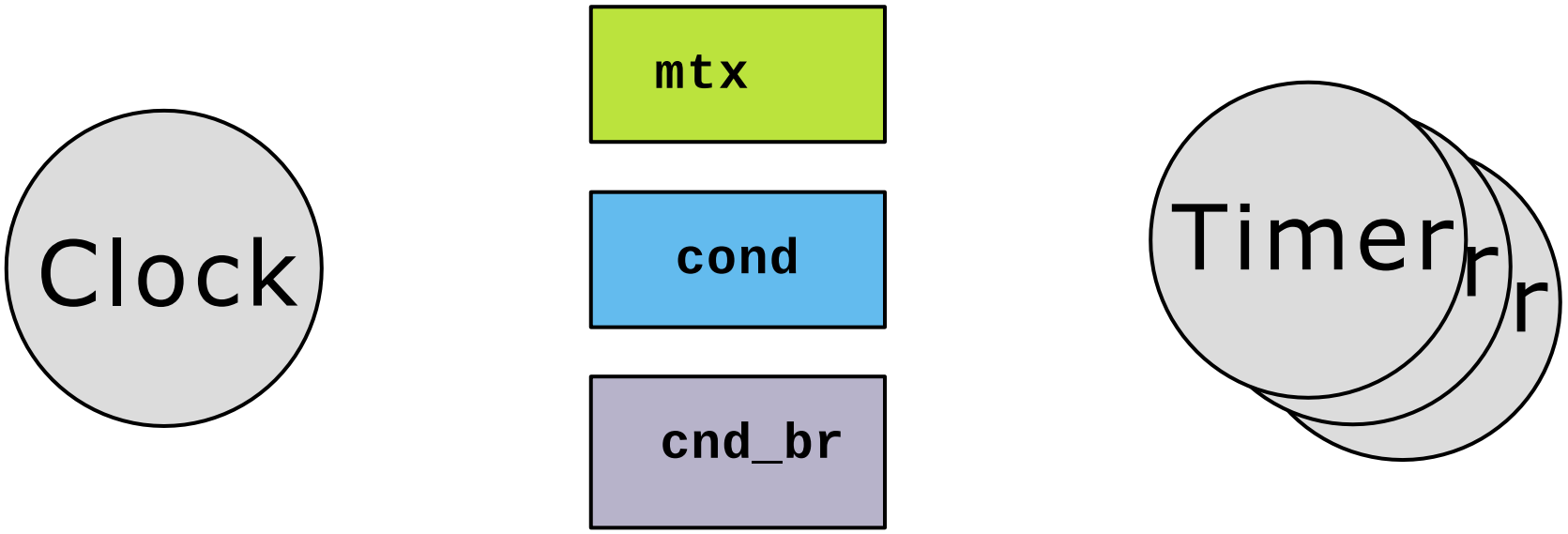
Conditional mutex




```
while(1){
  mutex_lock(&mtx);
  while(done < timers_num)
    cond_wait(&cond, &mtx);
  ...
  done = 0;
  cond_broadcast(&cnd_br);
  mutex_unlock(&mtx);
}
```

```
mutex_lock(&mtx);
while(1){
  done++;
  ...
  cond_signal(&cond);
  cond_wait(&cnd_br, &mtx);
}
```

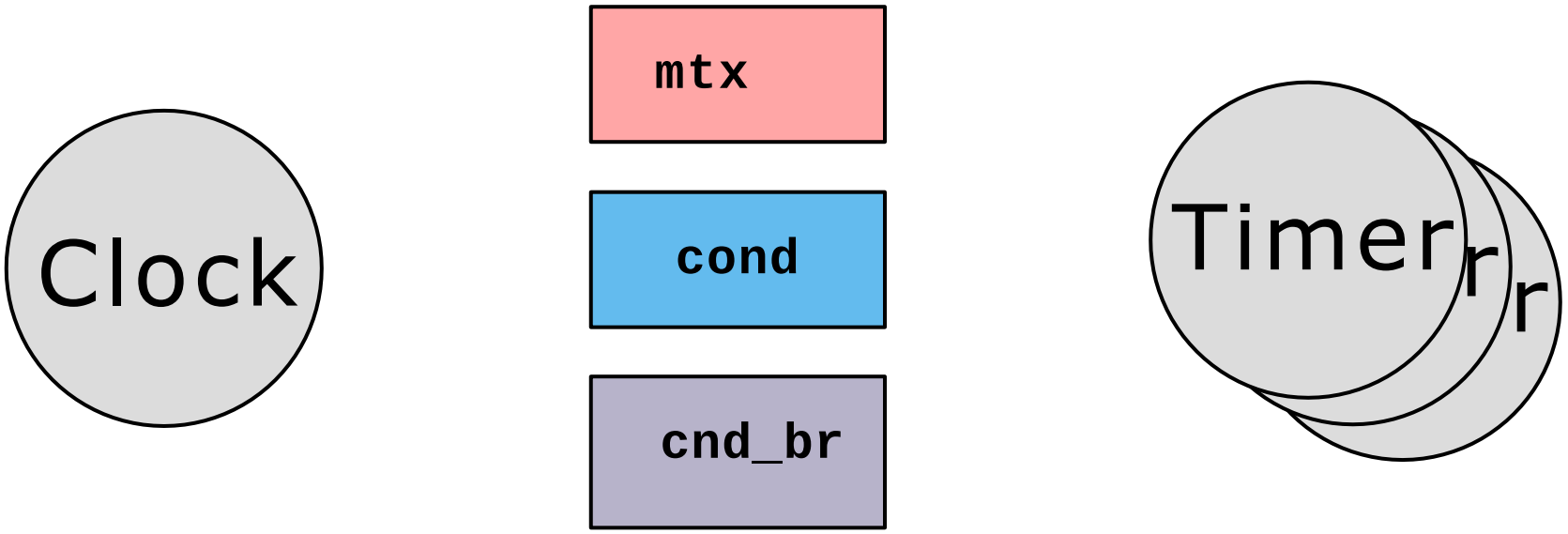
Conditional mutex




```
while(1){
  mutex_lock(&mtx);
  while(done < timers_num)
    cond_wait(&cond, &mtx);
  ...
  done = 0;
  cond_broadcast(&cnd_br);
  mutex_unlock(&mtx);
}
```

```
mutex_lock(&mtx);
while(1){
  done++;
  ...
  cond_signal(&cond);
  cond_wait(&cnd_br, &mtx);
}
```

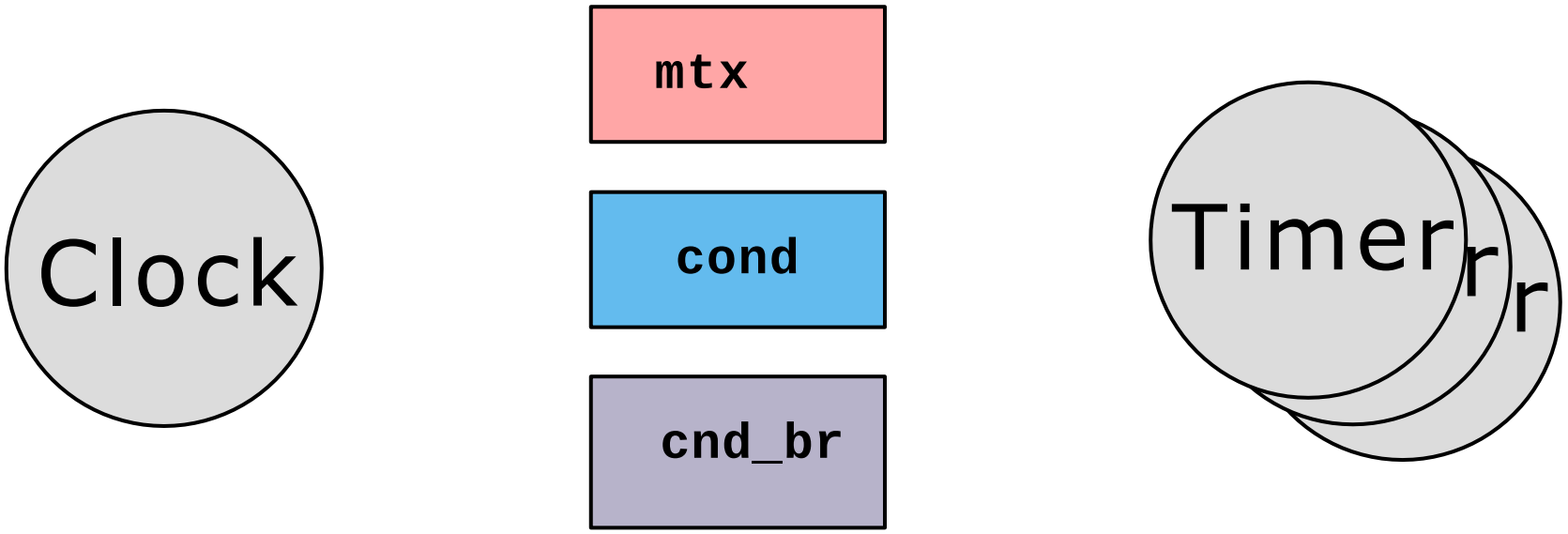
Conditional mutex




```
while(1){
  mutex_lock(&mtx);
  while(done < timers_num)
    cond_wait(&cond, &mtx);
  ...
  done = 0;
  cond_broadcast(&cnd_br);
  mutex_unlock(&mtx);
}
```

```
mutex_lock(&mtx);
while(1){
  done++;
  ...
  cond_signal(&cond);
  cond_wait(&cnd_br, &mtx);
}
```

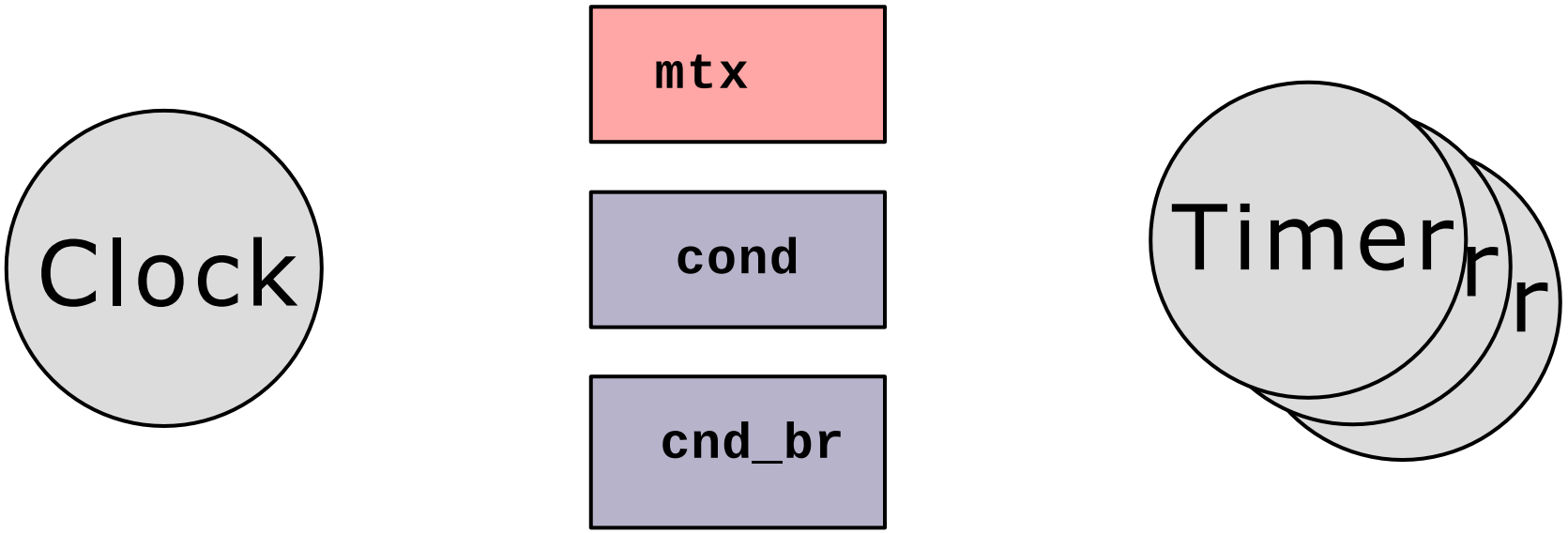
Conditional mutex




```
while(1){
  mutex_lock(&mtx);
  while(done < timers_num)
    cond_wait(&cond, &mtx);
  ...
  done = 0;
  cond_broadcast(&cnd_br);
  mutex_unlock(&mtx);
}
```

```
mutex_lock(&mtx);
while(1){
  done++;
  ...
  cond_signal(&cond);
  cond_wait(&cnd_br, &mtx);
}
```

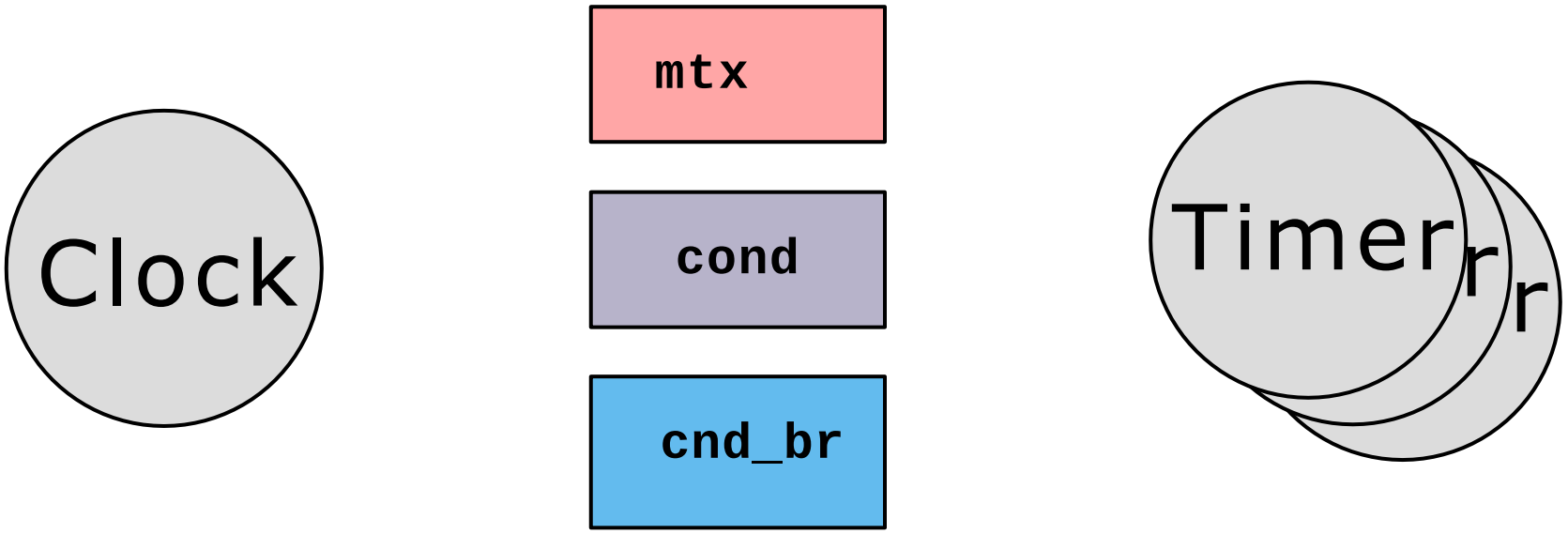
Conditional mutex




```
while(1){
  mutex_lock(&mtx);
  while(done < timers_num)
    cond_wait(&cond, &mtx);
  ...
  done = 0;
  cond_broadcast(&cnd_br);
  mutex_unlock(&mtx);
}
```

```
mutex_lock(&mtx);
while(1){
  done++;
  ...
  cond_signal(&cond);
  cond_wait(&cnd_br, &mtx);
}
```

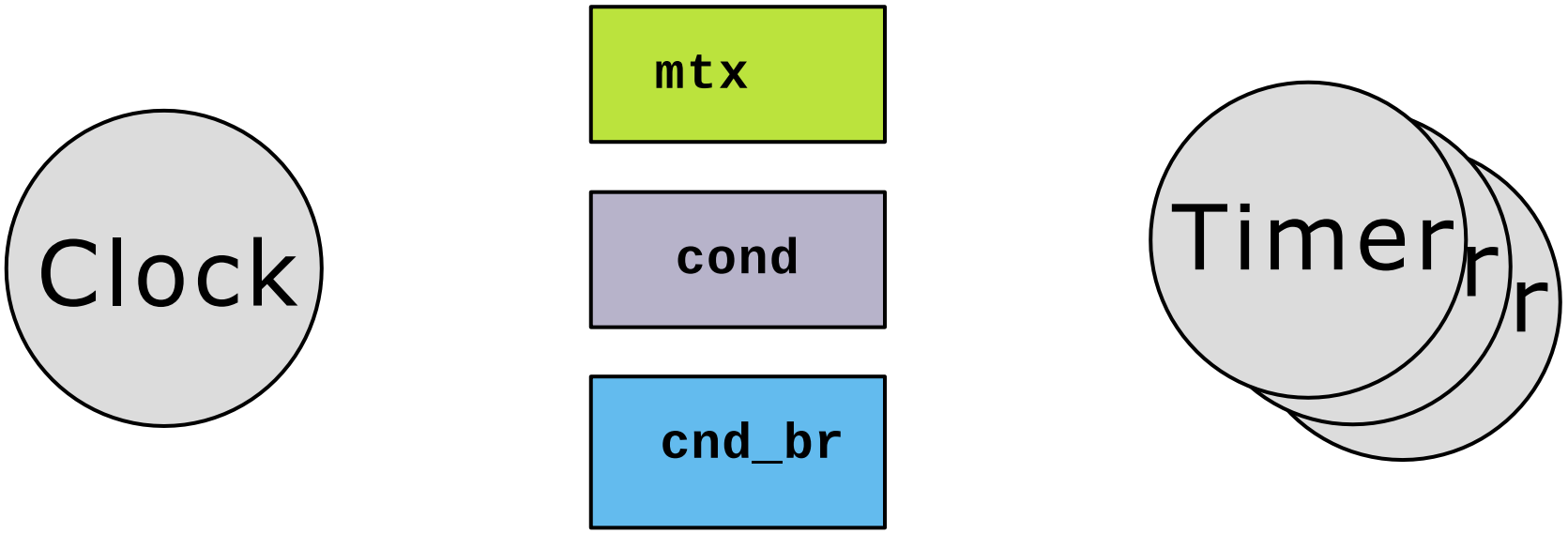
Conditional mutex




```
while(1){
  mutex_lock(&mtx);
  while(done < timers_num)
    cond_wait(&cond, &mtx);
  ...
  done = 0;
  cond_broadcast(&cnd_br);
  mutex_unlock(&mtx);
}
```

```
mutex_lock(&mtx);
while(1){
  done++;
  ...
  cond_signal(&cond);
  cond_wait(&cnd_br, &mtx);
}
```

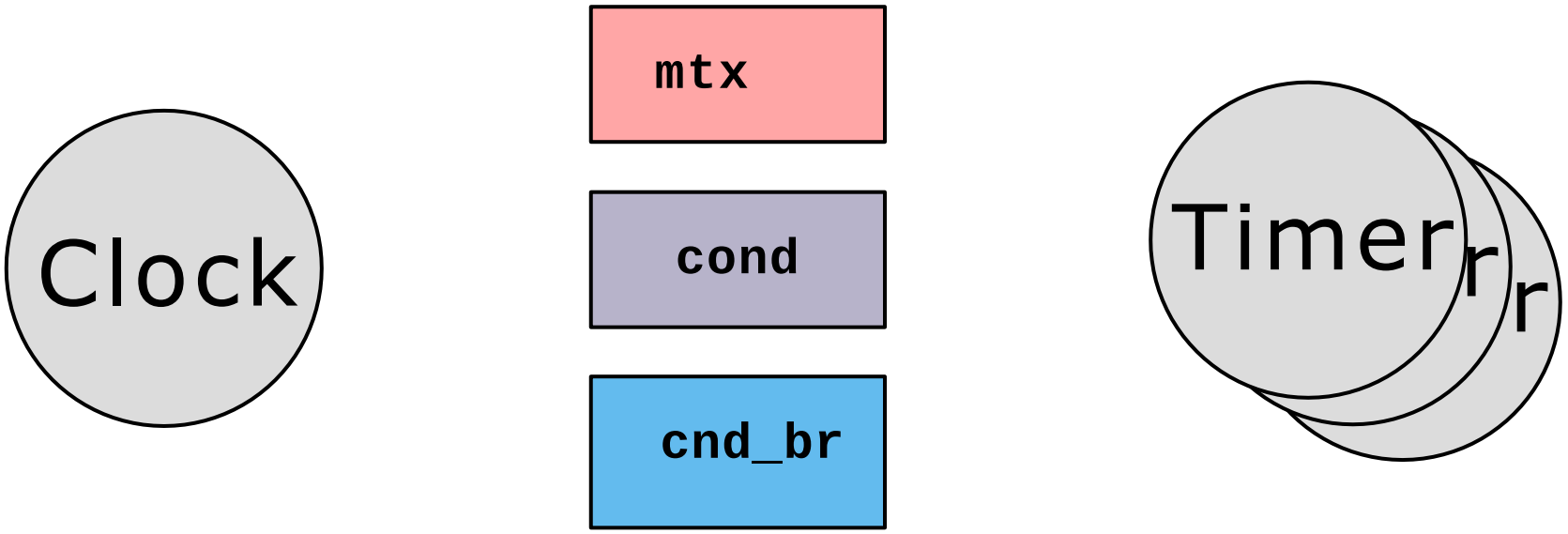
Conditional mutex




```
while(1){
  mutex_lock(&mtx);
  while(done < timers_num)
    cond_wait(&cond, &mtx);
  ...
  done = 0;
  cond_broadcast(&cnd_br);
  mutex_unlock(&mtx);
}
```

```
mutex_lock(&mtx);
while(1){
  done++;
  ...
  cond_signal(&cond);
  cond_wait(&cnd_br, &mtx);
}
```

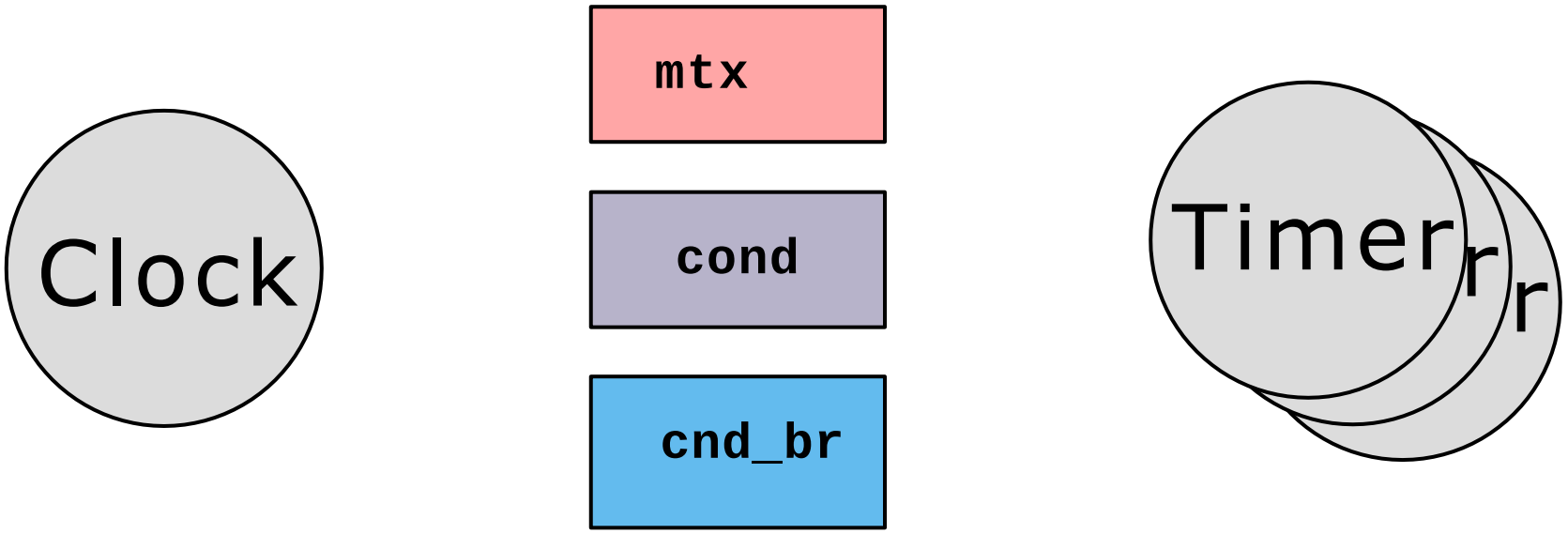
Conditional mutex




```
while(1){
  mutex_lock(&mtx);
  while(done < timers_num)
    cond_wait(&cond, &mtx);
  ...
  done = 0;
  cond_broadcast(&cnd_br);
  mutex_unlock(&mtx);
}
```

```
mutex_lock(&mtx);
while(1){
  done++;
  ...
  cond_signal(&cond);
  cond_wait(&cnd_br, &mtx);
}
```

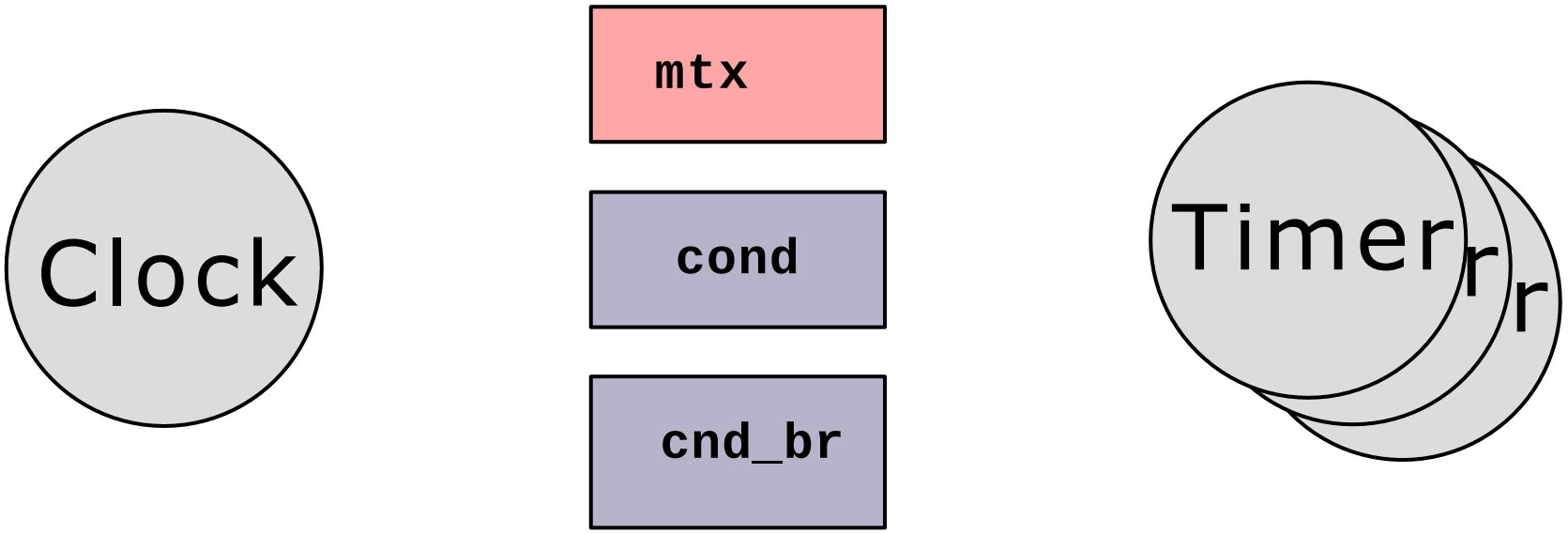
Conditional mutex




```
while(1){
  mutex_lock(&mtx);
  while(done < timers_num)
    cond_wait(&cond, &mtx);
  ...
  done = 0;
  cond_broadcast(&cnd_br);
  mutex_unlock(&mtx);
}
```

```
mutex_lock(&mtx);
while(1){
  done++;
  ...
  cond_signal(&cond);
  cond_wait(&cnd_br, &mtx);
}
```

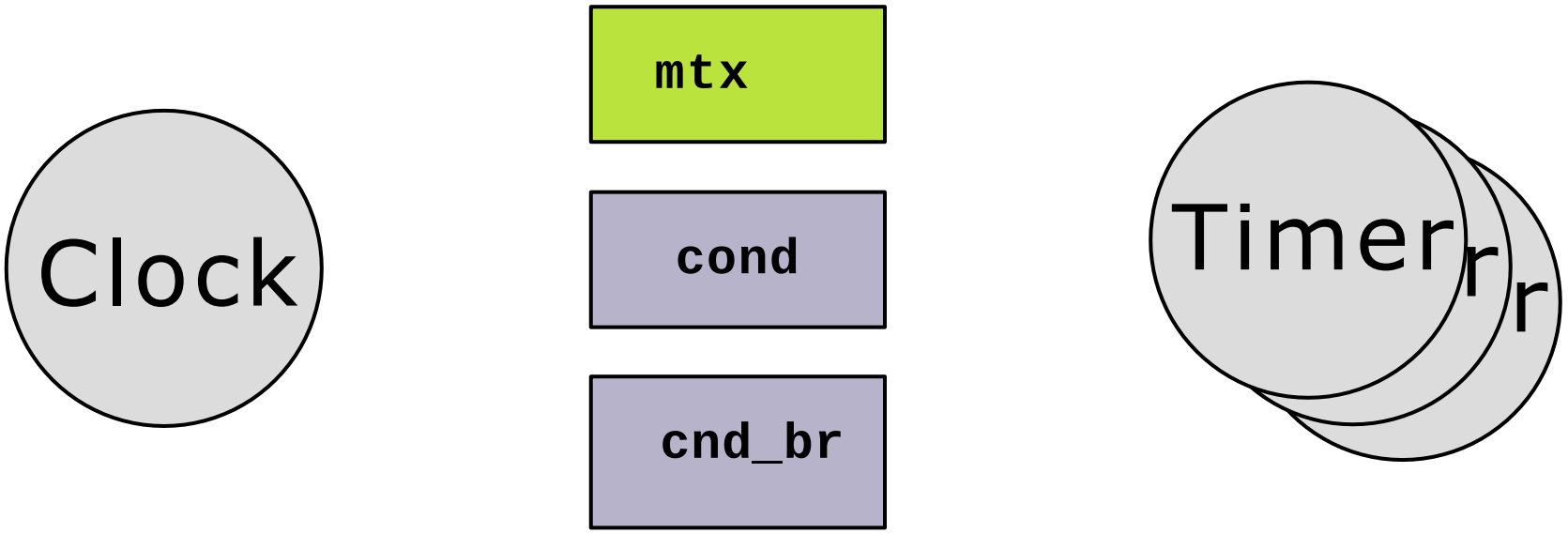
Conditional mutex




```
while(1){
  mutex_lock(&mtx);
  while(done < timers_num)
    cond_wait(&cond, &mtx);
  ...
  done = 0;
  cond_broadcast(&cnd_br);
  mutex_unlock(&mtx);
}
```

```
mutex_lock(&mtx);
while(1){
  done++;
  ...
  cond_signal(&cond);
  cond_wait(&cnd_br, &mtx);
}
```

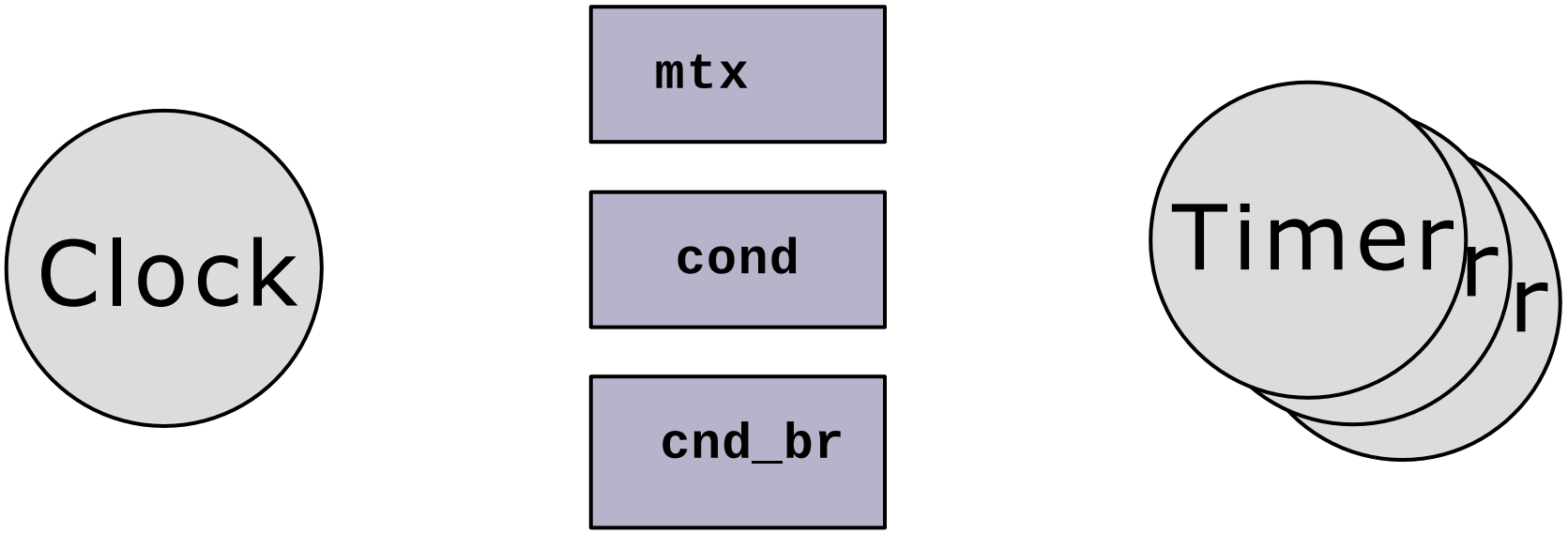
Conditional mutex




```
while(1){
  mutex_lock(&mtx);
  while(done < timers_num)
    cond_wait(&cond, &mtx);
  ...
  done = 0;
  cond_broadcast(&cnd_br);
  mutex_unlock(&mtx);
}
```

```
mutex_lock(&mtx);
while(1){
  done++;
  ...
  cond_signal(&cond);
  cond_wait(&cnd_br, &mtx);
}
```

Conditional mutex




```c
while(1){
  mutex_lock(&mtx);
  while(done < timers_num)
    cond_wait(&cond, &mtx);
  ...
  done = 0;
  cond_broadcast(&cnd_br);
  mutex_unlock(&mtx);
}
```

```c
mutex_lock(&mtx);
while(1){
  done++;
  ...
  cond_signal(&cond);
  cond_wait(&cnd_br, &mtx);
}
```