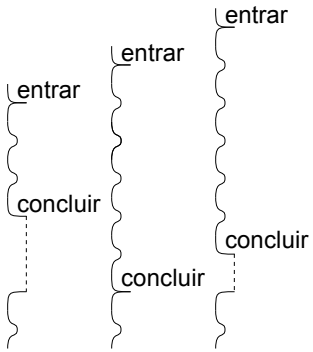
Los funcionarios de una organización pueden asistir libremente a una reunión entrando a la sala en donde se realiza, pero solo pueden salir si todos los asistentes están de acuerdo en concluir la reunión. Los funcionarios son representados por *pthreads*. Programe las siguientes funciones que serán invocadas por los funcionarios:

- `Reunion *nuevaReunion()`: entrega una nueva reunión a la cual los funcionarios pueden asistir.
- `void entrar(Reunion *r)`: ingresa a la reunión *r*.
- `void concluir(Reunion *r)`: vota por finalizar la reunión *r*. Concluir se bloquea en espera hasta que todos los funcionarios que asistieron a la reunión hayan votado por finalizar la reunión.

La siguiente figura muestra con un diagrama de threads un ejemplo de uso de una reunión.



Observe que concluir retorna solo cuando los 3 funcionarios que entraron a la reunión votaron por concluirla. Pueden haber distintas reuniones con distintos asistentes y no deben interferir entre sí (es decir no puede usar variables globales en su programación). Resuelva este problema de sincronización programando las funciones pedidas en el archivo *reunion.c*. Además Ud. debe declarar la estructura Reunion de la siguiente manera:

```
typedef struct reunion {
  … // declare acá un mutex y una condición y los contadores que
    // Ud. necesitará para resolver el problema de sincronización
} Reunion;
```

Requerimientos

Su tarea debe aprobar el test de prueba incluido en el archivo *test-reunion.c*.

Además se inspeccionará su tarea para verificar el uso correcto del mutex y la condición. Ud. debe evitar *dataraces*, *deadlocks* y ciclos de *busy-waiting*.

Recursos

Baje de material docente el archivo *t3.zip*. Contiene los archivos (i) *test-reunion.c* que verifica que su tarea funciona de acuerdo al enunciado, (ii) *reunion.h* con los encabezados de las funciones que Ud. debe programar y que Ud. debe incluir en el archivo *reunion.c*, (iii) *Makefile* para compilar su tarea, y (iv) *reunion.c.plantilla* que corresponde a la plantilla del archivo *reunion.c* que Ud. debe crear para resolver el problema.

Compile y ejecute su tarea con los comandos:

```
$ make test-reunion
$ ./test-reunion
```

El comando make usa las reglas incluidas en el archivo *Makefile* para compilar todos los archivos. El programa lo felicitará si pasa todos los tests.

Entrega

Ud. debe entregar mediante U-cursos el archivo *reunion.c*. No incluya otros archivos por favor. No se aceptarán tareas que no pasen los tests de unidad suministrados. Se descontará medio punto por día de atraso (excepto sábados, domingos y festivos).