




# Condition Variables




Supponiamo che un thread debba aspettare che una certa variabile diventi  $>10$ . Sicuramente **non** dobbiamo scrivere:

```
while(true) {  
    if(v>10) break;  
}
```

in quanto è un “busy waiting” estremamente inefficiente


Nota: i semafori ci permettono già di aspettare fino a quando la variabile associata non diventa  $>0$



Le *condition variables* permettono di rendere più efficiente l'attesa facendo eseguire il test solamente quando ha senso farlo, cioè quando la variabile da testare viene modificata.

Questo non avviene automaticamente, dobbiamo programmarlo noi utilizzando anche un *mutex*

Vediamo un esempio *semplificato* di attesa/sblocco



Esempio di attesa:


```
pthread_mutex_lock(&m);          // blocco m
while (v <=10)
    // aspetto novita' attraverso c
    pthread_cond_wait(&c, &m);    // rilascia m temporaneamente
pthread_mutex_unlock(&m);        // rilascio m definitivamente
```

Esempio di (possibile) sblocco

```
pthread_mutex_lock(&m);          // blocco m
v += x;                          // aggiorno v
pthread_cond_signal(&c);         // avverto che v è cambiata
pthread_mutex_unlock(&m);        // rilascio m
```

Le operazioni su **c** devono essere fatte con il blocco su **m**

La **wait** rilascia **m** ma quando riparte lo riblocca



La libreria ci mette a disposizione la condition variable **c**, tocca a noi usarla in maniera corretta insieme al mutex **m** e alla variabile da testare **v** (la condizione da testare può coinvolgere più variabili)

**Visione alternativa:** i semafori sono meccanismi di attesa basati su una variabile intera che possiamo variare di  $\pm 1$ . Le condition variables sono un meccanismo di attesa puro, dobbiamo gestire noi le variabili con l'aiuto di un mutex.