PDP 07-04

Threads

Definizione

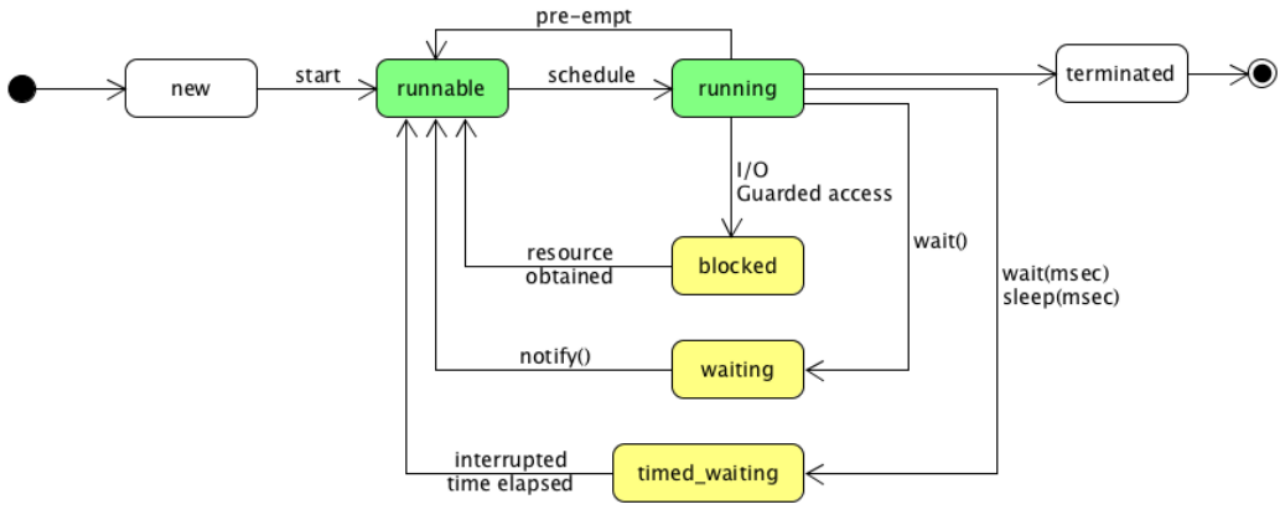
Struttura più piccola schedulabile e gestibile indipendentemente.

```
// Metodi utili
run()
start()
sleep()
interrupt()

// Numero fisso di thread riutilizzabili
ThreadSupplier()

// Interfacce
Executor() // Istanza runnable per i thread
ExecutorService() // Fornire metodi ed produzione
Future() // attesa di risultati pendenti e calcolo di un valore che
ritorna
Callable() // Invocazione dei task asincroni
```

Stati del thread



Virtual threads

- Gestiti sintatticamente dai thread
- Operazioni di:
  - unmounting
  - mounting

Sincronizzazione

```
synchronized
// Sezione critica = Accedono più processi
// Gestione alternativa con monitor

// Gestione dei thread
notify()
notifyAll()

// Ordine garantito dalla sintassi
final static ....
// Andare a gestirlo esplicitamente

// Lock

- sincronizziamo per creare un blocco implicito
- wait
        - lock
        - unlock

// Pattern per prendere e consumare risorse
- producer
- consumer

ReentrantLock // lock gestito manualmente
// thread in starvation = attesa infinita del lock

Condition
// separare l'accodamento in attesa dal possesso del lock che controlla
l'attesa
await()
signal()

// Semafori = gestione esplicita dei lock
```

```
public Semaphore(int permits, boolean fair)

// Fair = FIFO
release()
reducePermits()

InterruptedException()
IllegalArgumentException()
```

Esempi di gestione

```
// Gestione tradizionale

class MyClass {
public static synchronized void method1() {
// code here
}

public static void method2() {
synchronized(MyClass.class) {
// code here
}
}
}

// Gestione avanzata

class MyClass {
private final ReentrantLock lock = new ReentrantLock();

public void method1() {
lock.lock();
try {
// code here
} finally {
lock.unlock();
}
}

public void method2() {
if (lock.tryLock()) {
try {
// code here
```

```
} finally {
lock.unlock();
}
} else {
// code here
}
}
}
```

Thread-safe

Risorse accessibili senza esporre comportamento imprevedibile = non-deterministico
Questo avviene in gestione di molteplici dati (strutture = stream).

- Variabili atomiche
  - Uso di un contatore
  - Garantisce una modifica safe
  - Non blocca i thread esistenti
  - CAS (Compare and Swap)
    - Fare in modo che le operazioni del contatore siano consistenti
    - Support hardware
- Volatile
  - Lettura garantita dalla memoria principale
- Stateless
  - Garantire che le operazioni non salvino nulla
- Strutture dati concorrenti (garantiscono accesso singolo agli elementi della collezione)
  - ConcurrentHashMap
  - ConcurrentList
  - BlockingQueue
- Iteratori appositi
  - Spliterator
- Thread local
  - Memorizzazione per un thread specifico
  - Simili alle variabili globali
  - Stateful = posso riutilizzare dati precedenti

- Scoped variables
  - Alernativa a thread local
  - Parametri impliciti implementati in modo invisibile da thread figli