Passaggio sicuro di testimone - three way handshake

passare il testimone lasciandolo solo quando è stato già preso dal successivo

Passaggio sicuro di testimone - three way handshake (1)

Descrizione del problema :

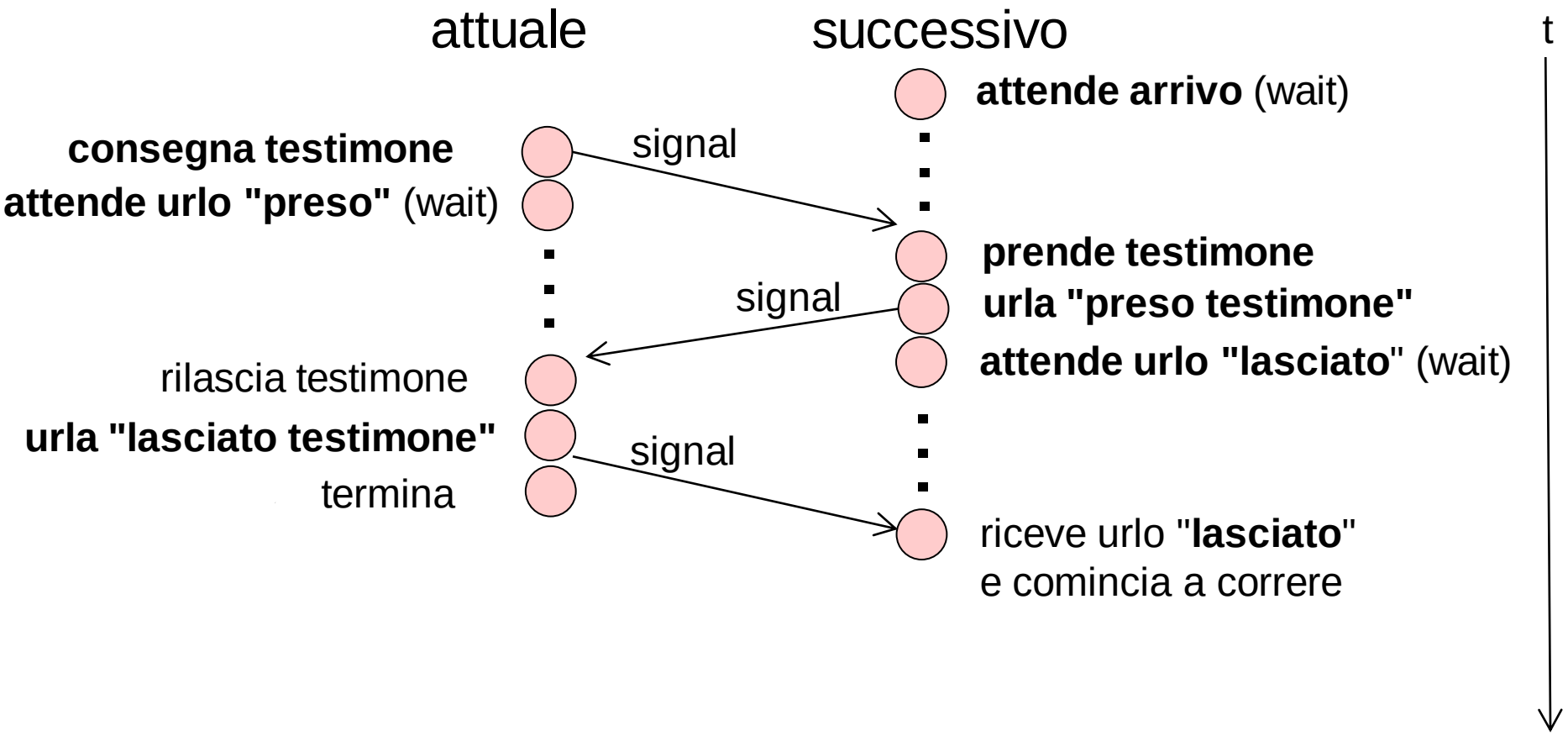
- Un gruppo di N atleti devono effettuare, uno alla volta, un giro di pista, portando con sè il bastoncino (testimone). Al termine del proprio giro di pista, il corridore **attuale** deve consegnare il testimone al **successivo** atleta che dovrà a sua volta effettuare il giro di pista.

Scopo della sincronizzazione : lasciare il testimone solo quando è stato già preso

- Guardiamo il dettaglio del passaggio di testimone, analizzando i requisiti:
  1) Il **successivo** deve **essere in attesa** dell'arrivo dell'attuale corridore, col la mano protesa all'indietro per ricevere il testimone.
  2) **L'attuale** effettua atomicamente due operazioni:
     2.1) **Appoggia il testimone nella mano del successivo**, per informarlo di afferrarlo, ma l'attuale non può lasciare il testimone fino a che il successivo non l'ha afferrato.
     2.2) Quindi **l'attuale si mette in attesa** di ricevere dal successivo l'urlo di conferma che il successivo ha afferrato il testimone.
  3) Il **successivo sente il testimone in mano** ed effettua atomicamente tre operazioni:
     3.1) **Afferra strettamente il testimone,** ma non può partire fino a che l'attuale non ha mollato la presa sul testimone, quindi
     3.2) **Urla all'attuale** per chiedergli di lasciare la presa e
     3.3) **Si mette in attesa di ricevere** a suo volta dall'attuale l'urlo che conferma di avere lasciato la presa sul testimone, per potersi mettere a correre.
  4) **L'attuale sente l'urlo del successivo** ed effettua atomicamente due operazioni:
     4.1) **lascia la presa** sul testimone e
     4.2) **urla al successivo di avere lasciato il testimone e quindi di partire**.
     Infine, si mette da parte. Ipotizziamo che per andar via ci metta poco tempo.
  5) Il **successivo riceve l'urlo dell'attuale che ha lasciato il testimone** e, non atomicamente, si mette a correre assumendo il ruolo di attuale.

Passaggio sicuro di testimone - three way handshake (2)

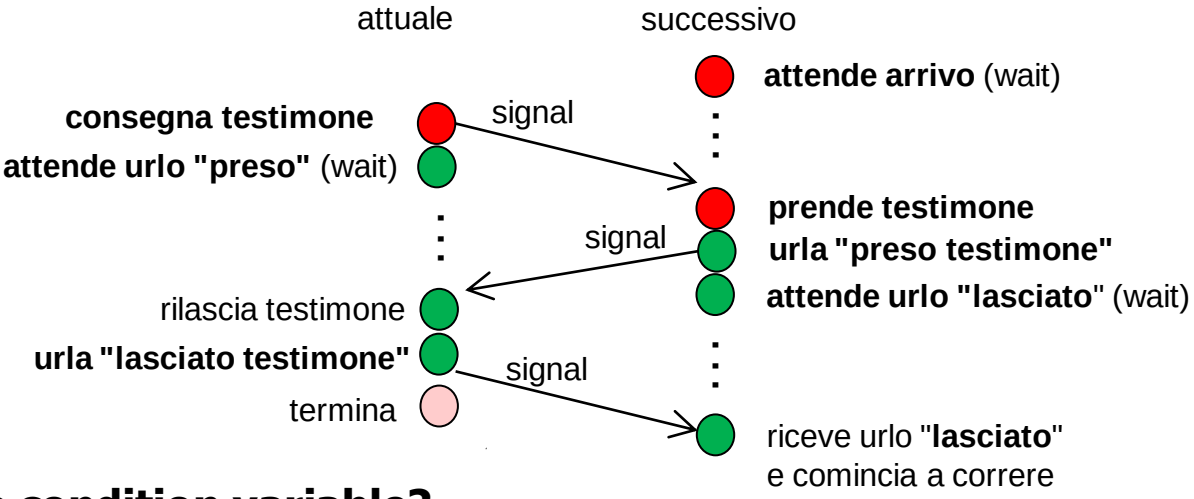
Schema delle azioni del passaggio di testimone (3 wait e 3 signal)



Implementazione Passaggio sicuro di testimone (1)

Quante condition variables? Due

1. **una mediante la quale tutti i corridori, che ancora non hanno corso, devono attendere l'arrivo del corridore che sta correndo ora.**
2. **l'altra per gli scambi di sincronizzazioni tra i soli due corridori attuale e successivo, dopo che l'attuale ha già appoggiato il testimone in mano al successivo.**



Perché NON usare una sola condition variable?

se usassi una sola cond var su quella coda sarebbero in attesa i più corridori in attesa di ricevere il testimone ed anche di volta in volta i due, attuale e successivo, che si stanno scambiando il testimone.

Rischierei di svegliare il thread sbagliato e dovrei fare troppi controlli e risvegli.

Potrei usarne più di due?

Si, potrei usare una condition variable diversa per ciascun punto di attesa (wait), nel nostro caso 3, ma è più efficiente usare meno cond var.

Occorre un compromesso tra efficienza e semplicità di scrittura del codice.

Implementazione Passaggio sicuro di testimone (2)

```
int turno=0;  mutex   cond_attesa_arrivo_precedente   cond_passaggio_testimone

void *corridore (void *arg)  {  int indice=*((int*)arg);  /*indice del corridore*/  int mioturno;
    pthread_mutex_lock( &mutex );  turno++;  mioturno=turno;
    if ( mioturno != 1 ) { /* non sono il primo, aspetto di ricevere testimone */
            /* attendo che il corridore precedente mi appoggi il testimone */
            pthread_cond_wait(  &cond_attesa_arrivo_precedente,&mutex);
            /* ho preso il testimone, urlo per dire che può lasciarlo */
            pthread_cond_signal( &cond_passaggio_testimone);
            /* per partire aspetto urlo di conferma di avere lasciato il testimone */
            pthread_cond_wait(  &cond_passaggio_testimone, &mutex);
    }
    pthread_mutex_unlock( &mutex ); /* parto a correre */
    sleep(1);           /* corro */
    if(mioturno<NUMCORRIDORI) {
            pthread_mutex_lock( &mutex );
            /* appoggio il testimone in mano al successivo ma non lo mollo */
            pthread_cond_signal(&cond_attesa_arrivo_precedente);
            /* attendo l'urlo del successivo prima di mollare il testimone */
            pthread_cond_wait(&cond_passaggio_testimone,&mutex);
            /* urlo al successivo che ho mollato e puo' partire */
            pthread_cond_signal( &cond_passaggio_testimone);
            pthread_mutex_unlock( &mutex );
    }
    pthread_exit(NULL);
}
```