SEMANTICA MESA – possibile implementazione delle funzioni delle variabili condizione. SVC è una chiamata al kernel (una supervisor call); la variabile ***lock*** usata è ***L***, e la ***variabile condizione*** è ***varcond***; ***insert(a,b)*** inserisce a nella coda b, con una politica non specificata; ***extract(coda)*** estrae un elemento da coda e lo restituisce. Se coda è vuota, restituisce NULL.

varcond.wait(L)
{lock.release(L); //rilascio della lock: uscita dalla sezione critica
SVC(insert(run,varcond);scheduler(ready)); // questa SVC inserisce il thread che ha eseguito la wait (che è run), nella coda della variabile condizione, ed attiva lo scheduler per scegliere un nuovo thread da eseguire scelto tra quelli della coda ready, quindi un nuovo run, un nuovo thread da mettere in esecuzione
lock.acquire(L); //acquisizione della lock: il thread che ha fatto la wait, quando viene riattivato, cioè quando è di nuovo messo in esecuzione, run, deve rientrare nella sezione critica
}

varcond.signal()
{SVC(insert(extract(varcond),ready); }// questa SVC può essere implementata nel modo seguente:

```
x=extract(varcond);
if x!=NULL
        {insert(x,ready)}
```

varcond.broadcast()
{SVC(broadcast(varcond),ready); }// questa SVC può essere implementata nel modo seguente:

```
x=extract(varcond);
while (x!=NULL)
        {insert(x,ready);
          x=extract(varcond);
        }
```

oppure semplicemente appendendo tutta la coda *varcond* in fondo alla coda *ready*, a seconda dello scheduler utilizzato

SEMANTICA HOARE – in questo caso la **wait** è come prima ma *senza* la *varcond.acquire(L)* finale (la signal "passa" la lock L al risvegliato); inoltre, la **signal** estrae con politica FIFO (la variabile condizione è gestita FIFO). *Non* è prevista la **broadcast** (ovviamente, visto come agisce la signal).