Monitores

Exclusion mutua simplificada

```
def accion():
  # realizar accion
```

Exclusion mutua

```
def entrar():
  while(not libre):
    wait(espera)
  libre=false

def salir():
  libre=true
  signal(espera)
```

Exclusion mutua hasta N elementos

```
def entrar():
  while(cant==N):
    wait(espera)
  cant++

def salir():
  cant--
  signal(espera)
```

Exclusion mutua hasta N elementos pero con Buffer limitado

```
def entrar(recurso):
  while(cant==N):
    wait(espera)
  cant++
  dep[libre]=recurso
  libre=(libre+1) mod N
  signal(hayRecurso)

def retirar(out recurso):
  while(cant==0):
    wait(hayRecurso)
  recurso = dep[ocupado]
  ocupado=(ocupado+1) mod N
  cant--
  signal(espera)
```

Exclusion mutua para escrituras pero no para lecturas

```
def entrar_leer():
  while(estaEscribiendo):
    wait(espera_lectura)
  cantLeyendo++

def liberar_leer():
  cantLeyendo--
  if(cantLeyendo==0):
    signal(espera_escritura)

def entrar_escribir():
  while(cantLeyendo>0 or estaEscribiendo):
    wait(espera_escritura)
  estaEscribiendo=true

def liberar_escribir():
  estaEscribiendo=false
  signal(espera_escritura)
  signal_all(espera_lectura)
```

Exclusion mutua respetando el orden de llegada

Passing the condition

```
libre=true # en un principio
def entrar():
  if(not libre):
    cantEspera++
    wait(espera)
  else:
    libre=false

def salir():
  if (cantEspera>0):
    cantEspera--
    signal(espera)
  else:
    libre=true
```

Exclusion mutua respetando el orden de prioridad

```
def entrar(id,prioridad):
  if(not libre):
    insertarOrdenado(colaEspera,id,prioridad)
    wait(espera[id])
  else:
    libre=false

def salir():
  if(empty(colaEspera)):
```

```
    libre=true
  else:
    signal(espera[pop(colaEspera)])
```

Respetando el orden por id

```
siguiente=0
def entrar(id):
  if(id!=siguiente):
    wait(espera[id])

def salir():
  siguiente++
  signal(espera[siguiente])
```

Timer

```
hora_actual=0
def demorar(intervalo,id):
  hora_despertar=hora_actual+intervalo1
  insertarOrdenado(dormidos,id,hora_despertar)
  wait(esperar[id])

def tick():
  hora_actual++
  masPrioridad=verPrimero(dormidos)
  while(masPrioridad<=hora_actual):
    signal(esperar[pop(dormidos)])
    masPrioridad=verPrimero(dormidos)
```

Rendezvous

Un solo Monitor

```
Monitor peluqueria():
  def llegar(id):
    push(colaPersonas,id)
    signal(hayPersonas)
    wait(termino)
    signal(clienteSalio)

  def atender(out id):
    if(empty(colaPersonas)):
      wait(hayPersonas)
    id=pop(colaPersonas)

  def salir():
    signal(termino)
    wait(clienteSalio)
```

Mas de un Monitor

```
Monitor Entrada():
  def entrar(out idAtencion):
    if(cantLibres==0):
      esperando++
      wait(espera)
    else:
      cantLibres--
    idAtencion=pop(colaAtencion)

  def proximo(idAtencion):
    push(colaAtencion,idAtencion)
    if(esperando>0):
      esperando--
      signal(espera)
    else:
      cantLibres++

Monitor Atencion[id:0..A-1]():
  def accion(recurso,out res):
    push(cola,recurso)
    signal(hayRecurso)
    wait(entrego)
    res=resCompartido
    signal(agarroEntrega)

  def esperar(out param):
    if(empty(cola)):
      wait(hayRecurso)
    param=pop(cola)

  def entregar(res):
    resCompartido=res
    signal(entrego)
    wait(agarroEntrega)
```

Semaforos

Exclusion mutua

```
mutex=1
Process Persona[id:0..A-1]():
  P(mutex)
  -- hacer algo
  V(mutex)
```

Exclusion mutua con N

```
mutex=N
```

```
Process Persona[id:0..A-1]():
  P(mutex)
  -- hacer algo
  V(mutex)
```

Exclusion mutua con una cola

```
espera=N
Process Persona[id:0..A-1]():
  P(espera)
  P(mutexCola)
  rec=pop(cola)
  V(mutexCola)
  usar(rec)
  P(mutexCola)
  push(cola,rec)
  V(mutexCola)
  V(espera)
```

Exclusion mutua con dos procesos con cantidad limitada N compartida

```
# PRIM<N, SEG<N
total=N
prim=PRIM
seg=SEG
Process Primero[id:0..P-1]():
  P(prim)
  P(total)
  V(prim)
  V(total)

Process Segundo[id:0..S-1]():
  P(seg)
  P(total)
  V(seg)
  V(total)
```

Exclusion mutua respetando orden de llegada

Passing the baton

```
Process Persona[id:0..P-1]():
  P(mutexLibre)
  if(not libre):
    push(cola,id)
    V(mutexLibre)
    P(espera[id])
  else:
    libre=false
```

```
    V(mutexLibre)
  HacerAlgo()
  P(mutexLibre)
  if not empty(cola):
    idLiberar=pop(cola)
    V(mutexLibre)
    V(espera[idLiberar])
  else:
    libre=true
    V(mutexLibre)
```

Exclusion mutua respetando orden por id estricto

```
Process Persona[id:0..P-1]():
  if id!=0:
    P(espera[id])
  HacerAlgo()
  if id!=P-1:
    V(espera[id+1])
```

Exclusion mutua con coordinador y orden de llegada

```
# variable con mutex en 1
# else en 0
Process Persona[id:0..P-1]():
  P(colaMutex)
  push(cola,id)
  V(colaMutex)
  V(hayGente)
  P(espera[id])
  HacerAlgo()
  V(termino)

Process Coordinador():
  while True:
    P(hayGente)
    P(colaMutex)
    idPersona=pop(cola)
    V(colaMutex)
    P(termino)
    V(espera[idPersona])
```

Exclusion mutua con coordinador y orden de llegada con N recursos

```
# variable con mutex en 1
# else en 0
# hayRecurso en 5
Process Persona[id:0..P-1]():
  P(colaMutex)
  push(cola,id)
```

```
  V(colaMutex)
  V(hayGente)
  P(espera[id])
  P(mutexRecurso)
  recurso=pop(colaRecurso)
  V(mutexRecurso)
  HacerAlgo(recurso)
  P(mutexRecurso)
  push(colaRecurso,recurso)
  V(mutexRecurso)
  V(hayRecurso)

Process Coordinador():
  while True:
    P(hayGente)
    P(colaMutex)
    idPersona=pop(cola)
    V(colaMutex)
    P(hayRecurso)
    V(espera[idPersona])
```

Barreras

Con un solo proceso

```
Process Persona[id:0..P-1]():
  P(mutex)
  cantPersonas++
  if(cantPersonas==P):
    for i in range(1,P):
      V(esperando)
  V(mutex)
  P(esperando)
```

Con dos procesos

```
Process Persona[id:0..P-1]():
  V(hayPersonas)
  P(esperando)

Process Coordinador():
  for i in range(1,P):
    P(hayPersonas)
  for i in range(1,P):
    V(esperando)
```

Exclusion mutua hacer M recursos con N procesos

```
Process Persona[id:0..P-1]():
  P(mutex)
  while(cantidad<M):
```

```
    cantidad++
    V(mutex)
    HacerAlgo()
    P(mutex)
  V(mutex)
```

Exclusion mutua hacer M recursos con N procesos en K grupos

```
Process Persona[id:0..P-1]():
  P(mutex[grupo])
  while(cantidad[grupo]<M):
    cantidad[grupo]++
    V(mutex[grupo])
    HacerAlgo()
    P(mutex[grupo])
  V(mutex[grupo])
```

Exclusion mutua hacer M recursos con N procesos habiendo coordinadores que los realizan

```
Process Persona[id:0..M-1]():
  coord=random(0,C-1)
  P(mutexCola[coord])
  push(colaPersonas[coord],id)
  V(mutexCola[coord])
  V(hayPersona[coord])
  P(esperarRespuesta[id])

Process Coordinador[id:0..C-1]():
  P(hayPersona[id])
  P(mutex)
  while(cantidad<M):
    cantidad++
    V(mutex)
    # checkear si liberar
    if cantidad==M:
      for i in range(1,C):
        V(hayPersona[i])
    # procesar
    P(mutexCola[id])
    idPersona=pop(colaPersonas[id])
    V(mutexCola[id])
    HacerAlgo()
    # avisar
    V(esperarRespuesta[idPersona])
    # esperas
    P(hayPersona[id])
    P(mutex)
  V(mutex)
```