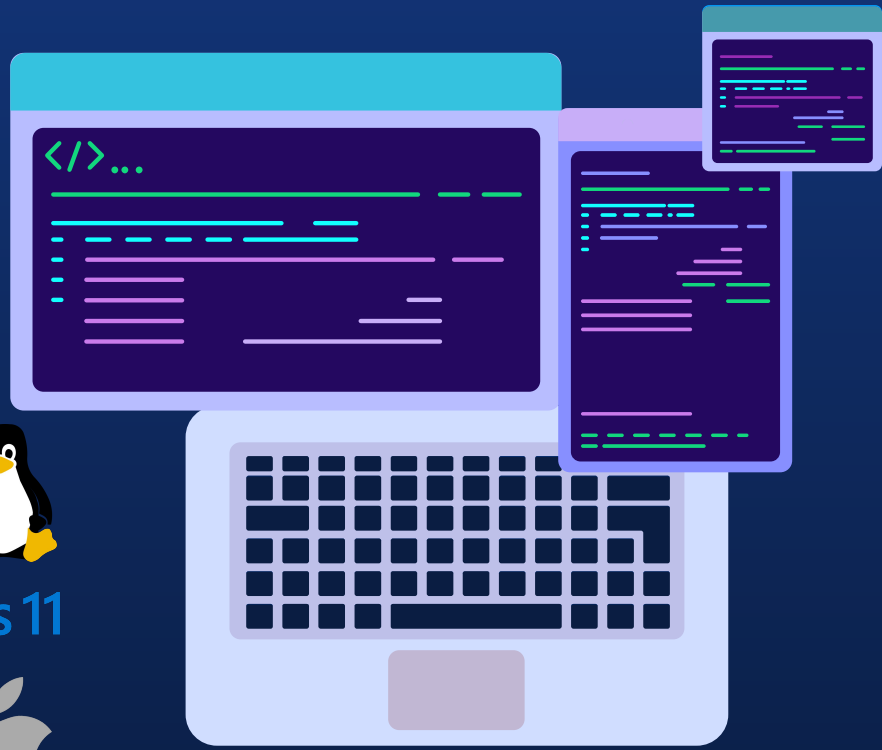
Universidad de O'Higgins

Ayudantia 3

Sistemas Operativos
COM4201



Linux™

Windows 11

Universidad de O'Higgins

Recordatorio

nSystem

nMain()	En vez del main normal, hace terminar las tareas que faltan antes de terminar
nEmitTask(int (*proc) (...))	Crea una nueva tarea, proc -> suele ser una función
nExitTask(int rc)	Termina una tarea, rc -> código de retorno
nWaitTask(nTask task)	Espera a que la tarea task termine
nExitSystem(int rc)	Termina la ejecución de todas las tareas, rc -> código de retorno

Monitores

nMakeMonitor( )	Construye el monitor
nEnter(nMonitor m)	Indica el inicio de la sección crítica
nExit(nMonitor m)	Indica el final de la sección crítica
nWait(nMonitor m)	Libera el monitor m y suspende la tarea que lo invoca hasta que otra tarea invoque nNotifyAll
nNotifyAll(nMonitor m)	Notifica a todos (los que están esperando) que se liberó el monitor m
nDestroyMonitor(nMonitor m)	Destruye el monitor m

Monitores estilo Hoare

nMakeCondition(nMonitor mon)	Construye la condición, asignandola al monitor mon
nDestroyCondition(nCondition cond)	Destruye la condición cond
nWaitCondition(nCondition cond)	Suspende una tarea como lo hace el nWait
nSignalCondition(nCondition cond)	Despierta a las tareas con la condición cond.

Problema 1 (Pregunta 2 - Control 1 – Semestre Primavera 2009)

Pregunta 2

En un pub del barrio universitario los estudiantes se reunen en la noche a tomar cerveza. Cada cierto tiempo, los estudiantes deben visitar el baño. El baño es amplio y admite un número ilimitado de estudiantes. El problema es que es uno solo y como es lógico damas y varones no deben compartir el baño. Se plantea la siguiente solución para evitar este inconveniente, en donde damas y varones se representan mediante tareas de nSystem que ejecutan los siguientes procedimientos:

```
nSem mutex; /* = nMakeSem(1); */
```

```
void varones() {
  while (no_amanezca()) {
    beber_cerveza();
    nWaitSem(mutex);
    ocupar_baño();
    nSignalSem(mutex);
  }
}
```

```
void damas() {
  while (no_amanezca()) {
    beber_cerveza();
    nWaitSem(mutex);
    ocupar_baño();
    nSignalSem(mutex);
  }
}
```

El dueño del pub se queja que las colas para ingresar al baño son muy largas, lo que perjudica su negocio de venta de cerveza. Por ello le pide a Ud. una nueva solución en donde un número ilimitado de varones puedan ingresar simultáneamente al baño (pero sin damas) o que un número ilimitado de damas puedan ingresar al baño (pero sin varones). Además le pide una solución justa (*fair*). Como herramienta de sincronización Ud. *debe* usar los monitores de nSystem.

Mensajes

nSend(nTask pid, *msg)	Enviar un mensaje (msg) a otro proceso destinatario (pid).
nReceive(nTask *pid, int t)	Espera a que llegue un mensaje y guarda en el pid local, el pid del remitente. Espera t segundos, usar número negativo para que espere indefinidamente.
nReply(nTask pid, int rc)	Responde al mensaje recibido, confirmación de haberlo recibido. Rc es el código de retorno.

Ejemplo de mensajes

Estructura

Semaforos	Monitores

nSem sem; // = nMakeSem(n);
 	
int fun(){
  …;
  nWaitSem(sem);
  // sección crítica
  nSignalSem(sem);
  …;
}
	nMonitor m; // = nMakeMonitor();
	
	int fun(){
	  …;
	  nEnter(m);
	  while(condicion){
	  nWait(m);
	  }
	  // sección crítica
	  nNotifyAll();
	  nExit(m);
	  …;
	}

Estructura

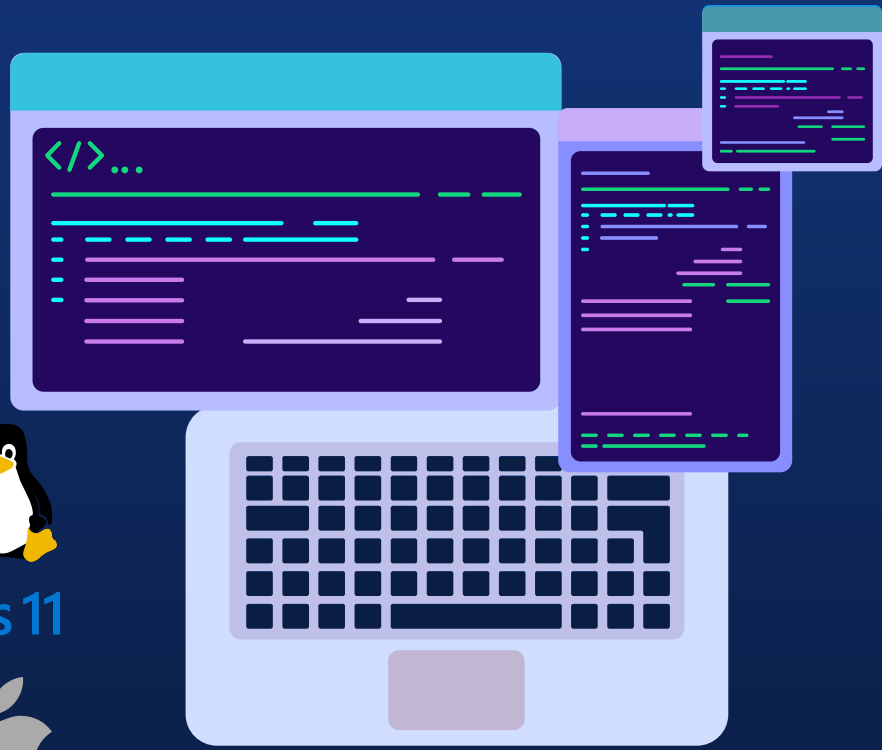
Monitores Hoare	Mensajes

nMonitro m;
nCondition = cond1, cond2;
 	
int fun(){
  …;
  nEnter(m);
  while(condicion){
  nWaitCondition(cond1);
  }
  // sección crítica
  nSignalCondition(cond2);
  nExit(m);
  …;
}
	int fun1(){
	  Item it;
	  …;
	  nSend(dest, &it);
	  …;
	}
	
	int fun2(){
	  nTask t;
	  Item *p_it= (Item*)nReceive(&t, -1);
	  …;
	  nReply(dest, rc);
	  …;
	}

Universidad de O'Higgins

Ayudantia 3

Sistemas Operativos
COM4201



Linux™

Windows 11