UNIVERSIDAD TECNICA
FEDERICO SANTA MARIA

# ELO 321
# Teoría de Sistemas Operativos
# Tarea #02
**Diciembre 08, 2018**

DEPARTAMENTO DE ELECTRONICA

**Semestre 2018 - 2**

**Profesor:** Ioannis Vourkas

**Ayudante:** Paulo Rivera

## Contenidos

Los contenidos principales a tratar son el uso y programación de *secciones críticas* y *sincronización* en la *comunicación* entre procesos.

## Objetivos

- Practicar el desarrollo de código C que crea procesos.
- Ejecutar programas que consideran conceptos de secciones críticas y sincronización de procesos.
- Familiarizarse con el desarrollo modular/incremental de código C en OS Linux.

---

## Descripción del trabajo a realizar

***Contexto***: Considere un único canal de comunicación bidireccional que une dos ciudades, Santiago y Valparaíso. En dicho canal, solo puede existir un mensaje en un momento dado. Se solicita generar el código que permite "simular" dicha comunicación utilizando comunicación entre procesos y conceptos de manejo de *secciones críticas* y *sincronización*.

***Especificaciones***: En su solución:

- El proceso padre debe generar dos (2) procesos hijos (Santiago y Valparaíso) que se comunicarán a través de un *pipe*.
- Cada ciudad enviará (y recibirá) un total de 5 mensajes diferentes.
- Cada ciudad enviará su mensaje *con cierta probabilidad*; es decir, no siempre se enviará un mensaje (use un generador de números aleatorios para conseguir esto.) Además, cada ciudad tendrá una cola que almacenará los mensajes que están por enviarse, y otra cola similar para la recepción de los mensajes que están por ser procesados.
- El envío de los mensajes debe simular *un retardo en el canal* proporcional al largo del mensaje enviado (es decir, generar un *delay* artificial mientras el proceso ocupa el canal de comunicación).

Su programa debe garantizar el uso correcto del canal de comunicación, evitando ***Deadlocks*** y ***Starvation***. Además, se debe registrar en un respectivo archivo `comm_log_file.txt` el avance de la ejecución del

programa, de tal manera que quede claro el envío y recepción correcta de los mensajes entre las dos ciudades, mostrando:

- el mensaje recién enviado
- el estado actual en la cola de envío (e.g. cuantos elementos tiene)
- el mensaje recibido y procesado
- el estado de la cola de recepción (e.g. cuantos elementos tiene)

El proceso padre debe **esperar que cada proceso hijo termine** con el envío y recepción de toda la información para terminar el programa.

---

**Indicación**

Las tareas se realizan **en grupos de 2 personas**. Cualquier acción que pueda beneficiar de forma injusta la calificación de su tarea está prohibida, incluyendo la presentación de cualquier componente que no es de su autoría, o la facilitación de esto para otros. Por supuesto, es aceptable discutir -*en líneas generales*- los métodos y resultados con sus compañeros, pero **se prohíbe explícitamente compartir soluciones de código. Utilizar código de internet que no es de su autoría, también es considerado plagio, a menos que se indique la fuente**. Por último, presten atención a las instrucciones de entrega, que pueden incluir políticas de nombre de archivos, codificación, y aspectos que se considerarán en la evaluación.

## Consideraciones y Formato de Entrega

- Utilice **/*comentarios*/** para documentar lo que se hace en cada etapa de su código.
- El código deberá estar escrito según el estándar de codificación **GNU** y estar perfectamente *indentado* con tabuladores, no espacios
- Toda tarea debe ser correctamente **compilada** y ejecutada **en el servidor Aragorn (**tareas que **no compilan** se evaluarán cualitativamente y **tendrán como máximo nota 54)**.
  - Para el cálculo de la nota final (**NF**) se aplicará la fórmula: **NF = (nota >= 30) ? nota : 0 ;**

- Para la entrega, envíe **un único archivo** comprimido (`.zip, .rar, .tar.gz`, etc.) que contenga sus archivos `.c, .h` utilizados más un archivo **`README.txt`** donde se especifican: los **datos de los integrantes del grupo** (nombres, apellidos, ROL USM, y emails), **y además** qué es cada archivo de su programa, cómo debe ser compilado y cómo debe ser ejecutado.

La tarea se entrega vía **AULA** hasta: **sabado 22 de diciembre 2018, 23:55hrs**