

# Proyecto de Sistemas Operativos 2025-30

## Simulación de Sistema de Reservas

### 1. Objetivos principales

- Resolver un problema utilizando procesos e hilos de la biblioteca POSIX.
- Emplear mecanismos de sincronización de procesos y comunicación de procesos usando pipes.
- Utilizar de forma correcta llamadas al sistema relacionadas con hilos y procesos.

### 2. Descripción general

En este proyecto se requiere implementar un sistema que permita hacer reserva en un parque privado por horas. El parque Berlín es muy pequeño, suelen ir muchas familias durante la época vacacional y los servicios dentro del parque colapsan. Por esta razón, los administradores han decidido realizar un sistema para que las familias puedan reservar horas en el parque, y así ellos puedan tener controlado un aforo máximo. Se requiere que el programa presente un sistema de reservas simulando las familias que entrarán a usar el servicio en un determinado día.

La aplicación requiere una arquitectura del tipo cliente/servidor. El servidor, llamado Controlador de Reserva, es un proceso que se encarga de las solicitudes de reserva del parque. Estas solicitudes son realizadas por varios Agentes de Reserva, que son los procesos cliente. Los procesos se comunicarán a través de pipes nominales. A continuación, se explican de forma general las funciones de estos componentes para posteriormente exponer detalles de su implementación.

#### 2.1. Servidor: Controlador de Reserva

- Recibe las solicitudes de los Agentes de Reserva y las autoriza o rechaza dependiendo de la cantidad de personas que ya ha reservado en las horas solicitadas. En algunos casos, es posible, que el sistema reserve en espacios de tiempo distinto al solicitado.
- El sistema simula el tiempo transcurrido, y al transcurrir una “hora”, realiza las siguientes acciones: sacar personas del parque porque se les

acaba su tiempo y autoriza la entrada de las personas que reservaron para la siguiente hora.

- Al finalizar el día, emite un reporte acerca de la ocupación del parque.

## 2.2 Cliente: Agente de Reserva

- Los Agentes son procesos que solicitan, al Controlador de Reservas, espacios para grupos familiares en un determinado día.

## 3. Implementación de los procesos

### 3.1 Controlador de Reserva

El servidor se invoca desde el terminal de la siguiente forma:

```
~$ ./controlador -i horaIni -f horaFin -s segHoras -t total -p pipeRecibe
```

Donde:

- **-i** flag o bandera que indica que a continuación se da el valor **horaIni**
- **horaIni** hora inicial para comenzar la simulación. El formato que se requiere es de 24 horas (1 - 24). Las horas de disponibilidad del parque o rango de simulación es desde las 7 hasta las 19 horas. Las horas se definen en números enteros. En caso de que la hora se encuentre fuera de ese rango de simulación, no debe ser válida.
- **-f** flag o bandera que indica que a continuación se da el valor **horaFin**
- **horaFin** hora final de la simulación. El formato que se requiere es de 24 horas (1 - 24). Las horas de disponibilidad del parque o rango de simulación es desde las 7 hasta las 19 horas. Las horas se definen en números enteros. En caso de que la hora se encuentre fuera de ese rango de simulación, no debe ser válida.
- **-s** flag o bandera que indica que a continuación se da el valor **segHoras**
- **segHoras** parámetro que indica el número de segundos de tiempo real que debe transcurrir una hora de simulación. En otras palabras, es el valor de una hora en segundos, para la simulación.
- **-t** flag o bandera que indica que a continuación se da el valor **total**
- **total** parámetro que indica la cantidad máxima de personas que pueden estar en el parque en cada hora (aforo permitido).
- **-p** flag o bandera que indica que a continuación se da el **pipeRecibe**

- **pipeRecibe** nombre de pipe por el cual los agentes envían información al Controlador de Reserva. En la **Figura 1**, **pipeRecibe** se ve representado como pipe1 (rectángulo de color verde).



### 3.1.1 Funciones del Controlador de Reserva

**A. Inicialización:** el proceso inicializa la hora de su reloj según el parámetro recibido (**horaIni**), inicializa sus estructuras de datos, realiza las instrucciones correspondientes para que el reloj de la señal que ha transcurrido una hora (dentro de los próximos **segHoras**), crea y abre el **pipeRecibe** para las comunicaciones con los agentes. Una vez realizadas estas tareas, se dispone a esperar, tanto la conexión de los agentes, como la señal del reloj (esto pudiera implementarse con **2 hilos**, tal y como se sugiere en la **Figura 1**). Mientras está esperando, pueden ocurrir los siguientes eventos:

- a. **Conexión de un agente:** Cada vez que se crea un Agente, éste le envía un mensaje inicial al controlador donde le escribe su nombre (ver parámetros de entrada del Agente). El controlador le devuelve la hora actual del sistema con base en la simulación que está realizando. **Esta comunicación inicial puede usarse también para que el Agente le envíe al controlador el nombre del pipe por donde se enviarán datos en el sentido Controlador->Agente y cualquier otra información que se considere pertinente.** Lo anterior se muestra en la **Figura 2**: El Agente le envía su nombre al controlador, así como el nombre del pipe que se usará para la comunicación en sentido Controlador->Agente (**PipeAgenteA** de color **morado**). El agente le devuelve la hora actual.

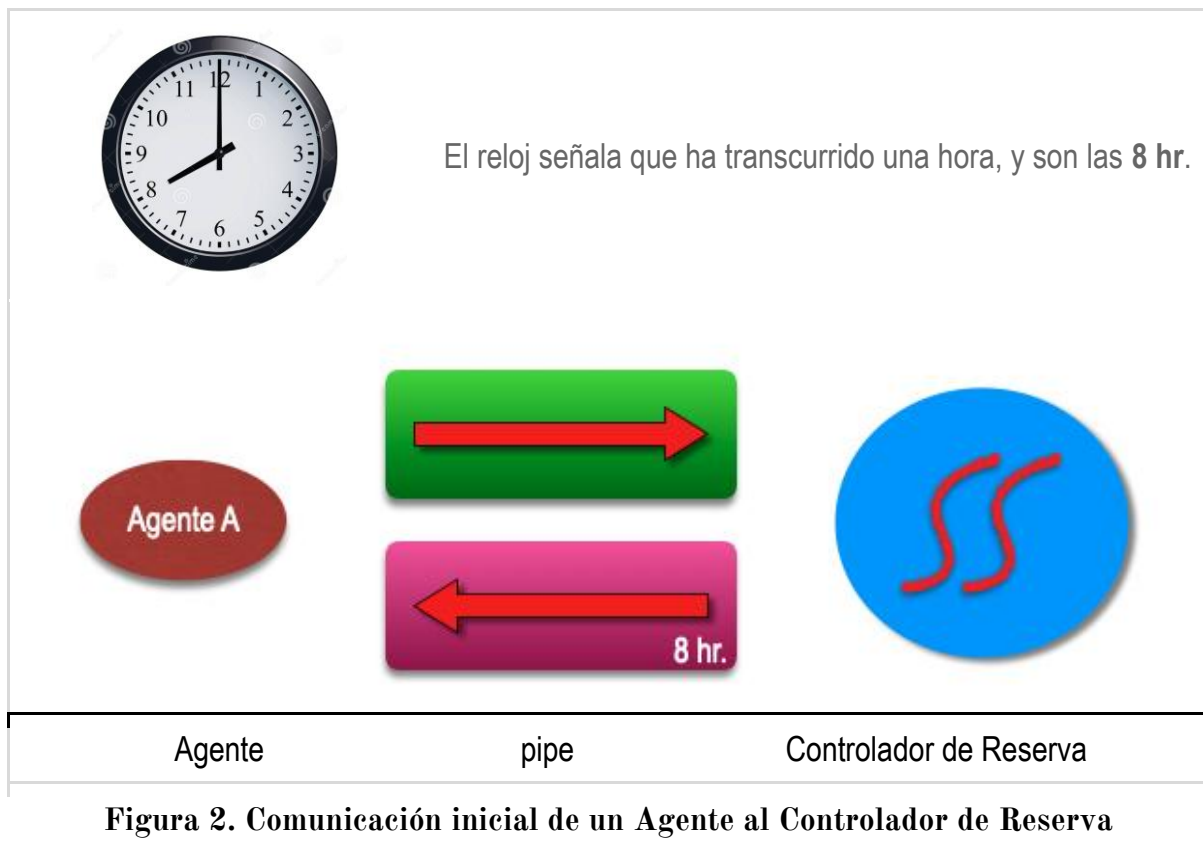


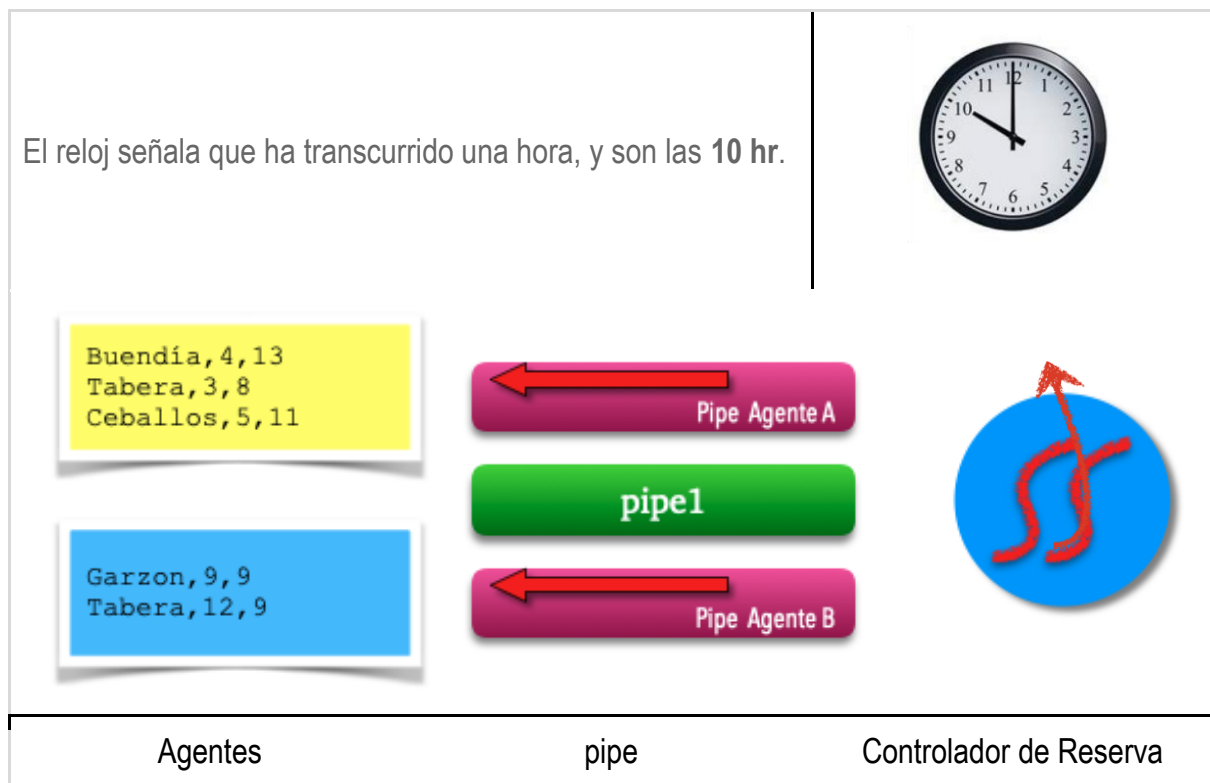
Figura 2. Comunicación inicial de un Agente al Controlador de Reserva

- b. **Recibir peticiones de los agentes:** las peticiones de los Agentes estarán relacionadas únicamente con la reserva para un grupo familiar de  $x$  personas que desean estar en el parque dos horas a partir de la hora  $w$ . La **Figura 3**, presenta un ejemplo con 2 Agentes enviando información al Controlador. El Controlador o un hilo del controlador, recibe la petición, revisa sus estructuras y puede devolver la siguiente respuesta al Agente:
- Reserva ok.** La solicitud es aprobada, las personas pueden entrar en el parque por 2 horas a partir de la hora solicitada. La hora solicitada se encuentra dentro del período de simulación.
  - Reserva garantizada para otras horas.** Para las horas solicitadas por el Agente ya no hay cupo en el parque, pero hay cupo en otras horas durante el periodo de simulación. En este caso, el controlador responde con una reserva de nueva hora para la familia.
  - Reserva negada por extemporánea.** En este caso el agente está solicitando una reserva para una hora que ya pasó. Al igual que en la situación anterior, si hay algún cupo de dos horas posteriores (dentro del rango de la simulación), se procede a realizar la reserva en horas diferentes a las solicitadas.

- iv. **Reserva negada, debe volver otro día.** El controlador negará la solicitud de reserva y no propondrá nuevas opciones cuando:
- La hora solicitada sea mayor a horaFin.
  - El agente ha solicitado una determinada hora y el controlador no encuentra disponible ningún bloque de dos horas dentro del periodo de simulación donde se pueda ubicar a la familia.
  - El número de personas incluidas en la reserva es mayor al aforo permitido.

El controlador debe imprimir las peticiones que va recibiendo en la pantalla (agente que las envía, hora solicitada y cantidad de personas). En la consola del Agente, también se deben poder observar las peticiones y sus respectivas respuestas.

- c. **Transcurre una hora:** Cada vez que ha transcurrido una “hora de simulación” el controlador imprime la hora actual en formato 1-24, también indica cuántas personas y qué familias salen y entran de la playa. Finalmente, avanza el reloj y coloca la alarma para la próxima hora.



**Figura 3. Envío de la información por 2 Agentes**  
(considerar que pueden estar conectados 2 o más Agentes.)

**B. Finalización:** el Controlador imprime un reporte al finalizar la simulación. El reporte del Controlador, según los parámetros iniciales establecidos, contiene:

- a. Horas pico u horas del día en las cuales el parque tuvo el mayor número de personas.
- b. Hora u horas del día en las cuales el parque tuvo el menor número de personas.
- c. Cantidad de solicitudes negadas
- d. Cantidad de solicitudes aceptadas en su hora
- e. Cantidad de solicitudes re-programadas

El Controlador debe terminar a los Agentes con una señal o se debe enviar un mensaje para que terminen.

## 3.2 Clientes: Agentes de Reserva

Los procesos clientes se invocan de la siguiente forma:

```
~$ ./agente -s nombre -a fileSolicitud -p pipeRecibe
```

Donde:

- **-s** flag o bandera que indica que a continuación se da el parámetro **nombre**
- **nombre** cadena de caracteres que identifica el nombre Agente.
- **-a** flag o bandera que indica que a continuación se da el parámetro **fileSolicitud**
- **fileSolicitud** es una cadena de caracteres que representa el nombre del fichero de entrada de donde cada agente toma las solicitudes de reserva. Es un fichero de texto, en donde cada línea representa una solicitud de reserva en Valores Separados por Comas (csv). El formato ejemplo es:

```
Zuluaga,8,10  
Dominguez,8,4  
Rojas,10,10
```

El primer valor es una cadena de caracteres para identificar la familia que hace la reserva, el segundo es la hora (7-19) representada por un número

entero, y el último valor de la línea es el número de personas solicitando la reserva. Cada reserva es por un total de dos horas.

El agente lee una línea, envía la solicitud al Controlador, espera la respuesta y la imprime. Luego, envía al Controlador la siguiente solicitud de reserva. Dado que el Agente, al registrarse con el controlador, recibe la hora actual de simulación, no debe enviar solicitudes con una hora anterior a la hora actual. La Figura 3 presenta un ejemplo con las solicitudes que va a enviar cada uno de los agentes.

- **-p** flag o bandera que indica que a continuación se da el **pipeRecibe**
- **pipeRecibe** nombre de pipe por el cual los agentes envían información al Controlador de Reserva. Es el mismo pipe que recibe el Controlador como parámetro en su creación.

### 3.2.1 Funciones y detalles del Agente

**A. Invocación y registro con el controlador:** los Agentes de Reserva se pueden iniciar en cualquier momento, después de que ha sido inicializado el Controlador de Reservas. Al iniciarse un Agente, abre el pipe de comunicación con el controlador (**pipecrecibe**) y le envía su nombre para que éste lo registre. Es posible que le envíe información adicional según el diseño que realice cada grupo, como por ejemplo, el nombre del pipe para recibir las respuestas del controlador (ver **Figura 2**)

**B. Lectura del archivo y solicitudes de reserva:** Una vez registrado con el Controlador, el Agente abre el archivo de entrada y ejecuta el siguiente algoritmo:

1. Leer la línea del archivo.
2. Validar la hora de reserva, la cual no debe ser inferior a la hora actual de simulación.
3. Enviar la solicitud y esperar la respuesta.
4. Imprimir la respuesta: positiva, reprogramación o denegada.
5. Esperar 2 segundos y volver al paso 1.

**C. Fin de Archivo:**

Cuando el archivo termina, el agente termina dando el siguiente mensaje:

Agente <**nombre**> termina.



## 4. Entrega del Proyecto

La entrega consiste en los códigos fuente del proyecto y un archivo Makefile, dentro de una carpeta comprimida (.zip, .tar). Se sustentará en clase, en la semana de la entrega configurada en Brightspace, que coincide con el cronograma del curso. Los proyectos que **NO** se entreguen para la fecha programada por el profesor, **NO** serán evaluados **NI** se podrán sustentar. Para la sustentación, el profesor indicará los archivos de entrada y los casos de prueba.

## 5. Observaciones adicionales

- El lenguaje de desarrollo es C, sistema operativo Linux y POSIX. Se deben validar las llamadas al sistema.
- Todos los estudiantes deben cargar el proyecto en BS, para que cada estudiante pueda tener acceso al código el día de la sustentación.
- Cualquier duda sobre el enunciado del proyecto debe consultarla con los profesores en forma oportuna. La comprensión del problema y su correcta implementación, según lo indica el enunciado, es parte de lo que se está evaluando.
- Los grupos pueden discutir e intercambiar ideas de forma verbal, **pero bajo ningún concepto pueden compartir código o resultados del informe**. Si se detecta copia en los productos entregados, los integrantes de los grupos serán citados a la Dirección de Carrera y el caso será elevado a las autoridades pertinentes de la Universidad.



## Sistema Simulador de Reservas de Parque

### Rúbricas Proyecto 2025-30

#### Sistemas Operativos

El objetivo del proyecto es la resolución por parte del estudiante de un problema en forma concurrente utilizando procesos, hilos y mecanismos de comunicación entre procesos. La **Tabla 1**, a continuación presenta los indicadores de desempeño.

Indicadores de Desempeño	Excelente	Bueno	Deficiente
Comprender el problema planteado. Respetar las especificaciones del enunciado.	Se resuelve el problema según el enunciado, respeta requerimientos de entrada, salida y todas las especificaciones para la implementación.	El estudiante resuelve el problema siguiendo las especificaciones de implementación, pero tiene fallas menores, básicamente en los requerimientos de entrada o de salida	El estudiante no siguió en su mayoría las especificaciones dadas para resolver el problema, no implementa alguna de funcionalidades o las entregas fueron incompletas
Crear una solución concurrente y funcional, al problema usando procesos e hilos.	El programa funciona perfectamente, según se especifica en el enunciado, para todos los casos de prueba. La concurrencia y los aspectos de comunicación síncrona y asíncrona se implementan correctamente.	No funcionan todos los casos de prueba.	No funcionan todos los casos de prueba o bien funcionan, pero hay errores conceptuales importantes en el uso de las llamadas al sistema para creación y/o comunicación síncrona o asíncrona. Los casos de prueba relacionados con el error conceptual, se invalidan o tienen calificación 0.
Realizar programas, que representan la solución al problema, de forma clara, modular, documentación satisfactoria y se manejan todos los posibles errores. Manejo apropiado de los recursos.	El código presenta documentación y estructura satisfactoria. Se ha tomado en cuenta las validaciones a las llamadas al sistema. No hay objeciones con respecto al estilo de programación. Se siguieron las indicaciones de la guía de estilo. Cierra archivos, elimina recursos temporales, usa solo los sleeps necesarios al momento de la apertura de pipes, entre otros.	El código presenta documentación con algunas deficiencias. Presenta estructura y hace validaciones a llamadas al sistema. Pequeñas fallas en estilo. Manejo adecuado de los recursos. Algunas instrucciones innecesarias.	Se presenta alguna falla importante en 2 o más de los siguientes aspectos: <ul style="list-style-type: none"> <li>• Documentación</li> <li>• Validación de llamadas al sistema</li> <li>• Estilo de Programación</li> <li>• Manejo adecuado de los recursos.</li> </ul>
Sustentación	El estudiante realiza correctamente y en el tiempo especificado la modificación solicitada.	El estudiante realiza parte de la funcionalidad requerida, pudiéndose encontrar en alguno de los siguientes casos:	El estudiante no realiza el ejercicio, codifica un porcentaje muy bajo de la solución (< 50%) o lo poco que realiza está

Indicadores de Desempeño	Excelente	Bueno	Deficiente
		<ul style="list-style-type: none"> <li>• Escribió casi todo el código pero no puede mostrar su funcionamiento.</li> <li>• Escribió al menos el 50% del código requerido para la solución</li> </ul>	incorrecto.

**Tabla 1.** Indicadores de desempeño del proyecto Simulador de Reservas de Parques.

La **Tabla 2** a continuación, presenta la guía de valoración para cada uno de los niveles de la Rúbrica

Actividad	Porcentaje	Excelente	Bueno	Deficiente
Comprender el problema planteado. Respetar las especificaciones del enunciado.	<b>3</b>	<b>3</b>	<b>2</b>	<b>&lt;2</b>
Solución del problema usando Procesos o hilos (ejecución)	<b>8</b> Calificación según casos de prueba teniendo en cuenta que, si tiene un error conceptual importante, según lo indicado en la tabla anterior, se penalizará todos los casos de prueba relacionados.			
Realizar programas, que representan la solución al problema, de forma clara, modular, documentación satisfactoria y se manejan todos los posibles errores. Manejo apropiado de los recursos.	<b>3</b>	<b>3</b>	<b>(3-2]</b>	<b>&lt;2</b>
Sustentación	<b>6</b>	<b>6</b>	<b>3</b>	<b>[1,0]</b>
<b>Total (20%)</b>				

**Tabla 2.** Guía de valoración para cada uno de los niveles de cada rúbrica.