**Proyecto Sistemas Operativos**



Sistema de Noticias (Patrón Publicador/Suscriptor)

**Profesor:** John Corredor Franco

Brayan Fajardo

Noviembre de 2024

**1. Introducción**

**Objetivo del Proyecto**:  
El objetivo de este proyecto es implementar un sistema de noticias utilizando el patrón Publicador/Suscriptor en el lenguaje de programación C, aprovechando la comunicación entre procesos en un sistema Unix/Linux a través de pipes nominales.  
Este proyecto ha sido elegido para ilustrar cómo el patrón Publicador/Suscriptor facilita la comunicación entre procesos. Es especialmente útil en sistemas donde los emisores no necesitan conocer a los receptores, lo cual es ideal para sistemas distribuidos.

**Resumen del Proyecto**:  
El sistema simula una plataforma de noticias donde los publicadores envían noticias de diversas categorías al Sistema de Comunicación (SC), que actúa como intermediario. Los suscriptores pueden elegir los tópicos de interés y recibir solo las noticias relevantes a esos temas. El sistema gestiona los mensajes entre publicadores y suscriptores, asegurando que cada suscriptor reciba únicamente los mensajes que le interesan.

**2. Descripción del Sistema  
  
Componentes del Sistema:  
  
Publicador:**Lee noticias de un archivo y las envía al Sistema de Comunicación (SC) a intervalos de tiempo definidos.  
 **Suscriptor:**Se suscribe a uno o más tópicos y recibe las noticias relevantes desde el SC.  
 **Sistema de Comunicación (SC):**Recibe noticias de los publicadores y gestiona las suscripciones de los suscriptores. Filtra y envía las noticias adecuadas.

**Patrón Publicador/Suscriptor**:  
El patrón Publicador/Suscriptor es un modelo de comunicación donde los publicadores (emisores de mensajes) envían eventos o datos sin saber quién los recibirá. Los suscriptores se registran en el sistema indicando los eventos o datos que les interesan, y reciben solo aquellos que coinciden con su interés.

**Componentes del Sistema**:

1. **Publicador**:
   * Funcionalidad: Lee noticias de un archivo y las envía al SC a intervalos de tiempo (timeN).
   * Invocación: ./publicador -p pipePSC -f file -t timeN.
   * Ejemplo de invocación: ./publicador -p pipePSC -f noticias\_p1.txt -t 2.
2. **Suscriptor**:
   * Funcionalidad: Se suscribe a uno o más tópicos y recibe las noticias relevantes desde el SC.
   * Invocación: ./suscriptor -s pipeSSC.
   * Ejemplo de invocación: ./suscriptor -s pipeSSC, luego ingresar tópicos como A C.
3. **Sistema de Comunicación (SC)**:
   * Funcionalidad: Recibe noticias de los publicadores y gestiona las suscripciones de los suscriptores. Filtra y envía las noticias a los suscriptores adecuados.
   * Invocación: ./sistema -p pipePSC -s pipeSSC -t timeF.
   * Ejemplo de invocación: ./sistema -p pipePSC -s pipeSSC -t 10.

**Esquema de Comunicación**:  
El sistema utiliza dos pipes nominales para la comunicación:

* **pipePSC**: Para enviar noticias desde los publicadores al SC.
* **pipeSSC**: Para registrar suscripciones y enviar noticias desde el SC a los suscriptores.

**3. Plan de Pruebas**

**Objetivo de las Pruebas**:  
Asegurar que el sistema funcione correctamente distribuyendo noticias, manejando suscripciones de acuerdo a los tópicos, y finalizando ordenadamente.  
Los casos de prueba se diseñan para garantizar la estabilidad del sistema, la correcta distribución de noticias, y el manejo adecuado de la finalización de procesos. Cada prueba tiene un resultado esperado claro para verificar el correcto comportamiento del sistema.

**Ambiente de Pruebas**:

* **Uso de Cocalc**
* **Lenguaje de programación**: ANSI C
* **Compilador**: GCC

**Casos de Prueba**:

| **ID** | **Caso de Prueba** | **Procedimiento** | **Resultado Esperado** |
| --- | --- | --- | --- |
| 1 | Iniciar el SC con pipes pipePSC y pipeSSC | Ejecutar ./sistema -p pipePSC -s pipeSSC -t 10 | SC se inicia y espera publicaciones y suscripciones sin errores. |
| 2 | Iniciar un Publicador con noticias de un archivo | Ejecutar ./publicador -p pipePSC -f noticias\_p1.txt -t 2 | El publicador envía las noticias del archivo noticias\_p1.txt a intervalos de 2 segundos. |
| 3 | Iniciar un Suscriptor y suscribirse a tópicos A y C | Ejecutar ./suscriptor -s pipeSSC, luego ingresar A C | El suscriptor se registra en SC y recibe noticias solo de los tópicos A (Arte) y C (Ciencia). |
| 4 | Enviar noticias de múltiples Publicadores | Iniciar otro publicador con ./publicador -p pipePSC -f noticias\_p2.txt -t 3 | El SC recibe noticias de ambos publicadores y las distribuye a los suscriptores correspondientes. |
| 5 | Confirmar mensajes de fin de emisión en los Suscriptores | Finalizar todos los publicadores y esperar timeF segundos en SC | Después de timeF segundos, cada suscriptor recibe el mensaje de "Fin de emisión de noticias". |
| 6 | Interrumpir un Suscriptor y verificar estabilidad en SC | Iniciar un suscriptor y luego detenerlo con Ctrl+C | SC maneja la desconexión sin errores y continúa operando para otros suscriptores y publicadores. |

**4. Desarrollo del Código**

**1. publicador.c**

* **Función principal**: El publicador lee noticias de un archivo y las envía al SC a intervalos definidos por timeN.
* **Validación de formato**: Verifica que cada noticia cumpla el formato especificado antes de enviarla.

**2. suscriptor.c**

* **Configuración de tópicos de interés**: Permite al usuario ingresar los tópicos al iniciar el proceso.
* **Recepción de noticias**: El suscriptor recibe solo las noticias de los tópicos a los que se suscribió.

**3. sistema.c**

* **Gestión de publicaciones y suscripciones**: Recibe noticias de los publicadores y empareja suscriptores de acuerdo a sus intereses.
* **Confirmación de envío**: Imprime en la consola cada noticia recibida y confirma el envío a cada suscriptor interesado.
* **Notificación de fin de emisión**: Envía un mensaje de "Fin de emisión de noticias" a todos los suscriptores después de que el SC ha esperado el tiempo timeF.

**5. Resultados y Observaciones**

**Resultados**: Las pruebas realizadas indican que:

* Los publicadores envían las noticias correctamente a intervalos definidos.
* Los suscriptores reciben solo las noticias correspondientes a los tópicos a los que están suscritos.
* El SC distribuye las noticias y confirma cada envío en la consola.
* Los suscriptores reciben el mensaje de "Fin de emisión de noticias" después de que el SC finaliza.

**Observaciones**:

* **Limitación**: Solo permite hasta 10 suscriptores simultáneos, dado que MAX\_SUBSCRIBERS está limitado a 10.
* **Posibles mejoras**: Cambiar la arquitectura de comunicación a colas de mensajes para gestionar más suscriptores y adaptarse mejor a sistemas distribuidos.  
  Para mejorar la escalabilidad, se recomienda utilizar sockets TCP/IP, lo cual permitiría manejar una mayor cantidad de suscriptores y distribuir la carga de trabajo en un entorno más amplio y distribuido.

**6. Conclusión**

El sistema de noticias basado en el patrón Publicador/Suscriptor cumple con los objetivos del proyecto, implementando un sistema de comunicación entre procesos donde los suscriptores reciben noticias de acuerdo a sus intereses en tiempo real. La arquitectura del patrón permite escalabilidad y flexibilidad en la gestión de suscripciones y noticias.

**Manual de Usuario**:  
  
Antes de ejecutar cualquier componente, asegúrese de compilar los archivos con GCC:  
  
gcc -o sistema sistema.c  
gcc -o publicador publicador.c  
gcc -o suscriptor suscriptor.c  
  
Para ejecutar el Sistema de Comunicación (SC):  
./sistema -p pipePSC -s pipeSSC -t 10  
El SC debe iniciarse antes de cualquier publicador o suscriptor.

* Para ejecutar el SC: ./sistema -p pipePSC -s pipeSSC -t 10
* Para ejecutar un Publicador: ./publicador -p pipePSC -f noticias\_p1.txt -t 2
* Para ejecutar un Suscriptor: ./suscriptor -s pipeSSC, luego ingresar los tópicos de interés (A, C, etc.)

**7. Referencias**

1. Kerrisk, M. (2010). *The Linux programming interface: A Linux and UNIX system programming handbook*. No Starch Press.
2. Linux man-pages project. (n.d.). *Pipe() - create pipe* (man page). <https://man7.org/linux/man-pages/man2/pipe.2.html>
3. GNU. (n.d.). *GNU C library documentation*. GNU Project. <https://www.gnu.org/software/libc/manual/>