Publisher/Subscriber IPC

Ottavia Belotti Riccardo Izzo

November 22, 2021

Contents

1 Project data

- Project supervisor(s): Federico Reghenzani
- Project team:

Last and first name	Person code	Email address
Belotti Ottavia	10657411	ottavia.belotti@mail.polimi.it
Izzo Riccardo	10599996	riccardo.izzo@mail.polimi.it

- Subdivision of development tasks:
  Ottavia Belotti:
  - subscriber_write: write function for subscribe device file
  - signal_nr_write: write function for signal_nr device file
  - endpoint_write: write function for endpoint device file
  - permissions of device files
  - concurrency

  Riccardo Izzo:
  - new_topic_write: write function for new_topic device file
  - release_file: function that release all the device files when unloading the module
  - subs_list_read: read function for subscribers_list device file

  - endpoint_read: read function for endpoint device file
  - ownership of device files

- Links to the project source code: `https://github.com/RiccardoIzzo/AOS-Publisher-Subscriber-IPC`

2 Project description

- What is your project about?
  The project consists in the implementation of a Linux kernel module that introduce an IPC mechanism based on the publisher/subscriber model, this IPC mechanism is not originally supported on Linux systems. The main idea is to have a publisher that writes a message on a topic, all the subscribers are notified through a POSIX signal that there is a message ready to be read.
- Why it is important for the AOS course?
  Inter-Process Communication mechanisms are essential in multitasking operating systems, they allow different processes to communicate and exchange information. The POSIX library is the one that provides IPC and is supported in the majority of Linux distribution. Moreover kernel modules are important because allow to extend the base kernel without rebuilding the kernel or rebooting the computer. Finally this project represents an excellent opportunity to put in practice what was studied in the AOS course.

2.1 Design and implementation

The linux kernel module has been written in C and has been tested on Ubuntu 20.04 LTS on x86 architecture. As every kernel module there are two functions that manage the loading and the unloading of the module:

- **psipc_init**: called when the module is loaded with *insmod* command, it initializes the list of topics and creates the new_topic device file.
- **psipc_exit**: called when the module is unloaded with *rmmod* command, it deletes the list of topics and unregisters all the device files created for every topic.

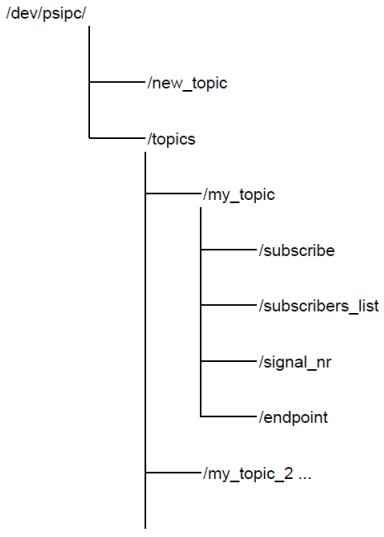

Figure 1: Psipc directory scheme

The module manages the following device files:

- **new_topic**: device file responsible of the creation of the topics, whenever the publisher writes here a new topic is created in the /dev/psipc/topics folder
- **subscribe**: device file that manages the registration of the subscribers, a process can write its PID here to subscribe itself to the topic. His pid is added to the list of PID associated to the topic.
- **subscribers_list**: device file that manages the list of subscribers pid, a process can retrieve the list of PID by reading it
- **signal_nr**: device file that manages the signal to send to the subscribers, the publisher can write here the numerical value associated to the POSIX signal. If the signal hasn't been set no message is sent to the subscribers.
- **endpoint**: the only device file that can be opened both in read and write mode. It is opened in write mode by the publisher that writes here the message, now a signal is sent to all the subscribers to notify them that they can read the message. To do so the subscribers open the device file in read mode and retrieve the last message. Note that a publisher cannot write a new message until all the notified subscribers have read the previous one.

3 Project outcomes

3.1 Concrete outcomes

The only artifact is **psipc_kmodule.c**, the linux kernel module. With the makefile is possible to compile it and obtain psipc_kmodule.ko, the loadable kernel object. To load the module simply run in the terminal "sudo insmod psipc_kmodule.ko", now it is successfully loaded.

3.2 Learning outcomes

Fundamentally we both learned how to develop an out-of-tree linux kernel module. We both understood that errors at kernel-level are not tolerated and usually result in a system crash, to avoid them is necessary the utmost attention to details. Another important thing we learned is the use of device files, we managed to let processes in user-space communicate with the module in kernel-space through them. We understood that each device file acts as an interface meaning that data is not actually stored in the file but copied from the user-space to the kernel-space and viceversa. Finally we learned how to use spinlocks and atomic operations in order to write concurrent code.

3.3 Existing knowledge

- **Advanced Operating Systems (AOS)**: understanding of the linux kernel, IPC mechanisms, device files, POSIX signals, kernel concurrency
- **Architettura dei Calcolatori e Sistemi Operativi (ACSO)**: basic understanding of operating systems
- **Fondamenti di Informatica**: C programming
- **Algoritmi e Principi dell'informatica (API)**: space and time complexity of the algorithms

3.4 Problems encountered

We had some problems setting the owner of the device files without using user-level known functions like *chown()*. At the end we solved it by directly editing the uid and gid fields in the inode struct associated to the device file. Another problem was how to manage input from user-space, we learned to use get_user() and put_user() functions.

4 Honor Pledge

We pledge that this work was fully and wholly completed within the criteria established for academic integrity by Politecnico di Milano (Code of Ethics and Conduct) and represents our original production, unless otherwise cited.
We also understand that this project, if successfully graded, will fulfill part B requirement of the Advanced Operating System course and that it will be considered valid up until the AOS exam of Sept. 2022.

Group Students' signatures