Project

Part #1: MDA-EFSM

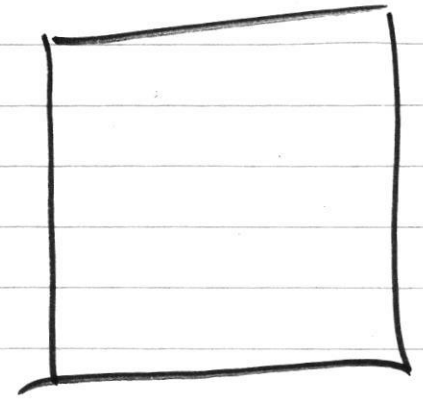
Deadline: Monday, April 1

Part #2: Detailed project description is posted

Pipes and Filters architecture

Filters: processing components.

Pipes: "transmit" data between filters.

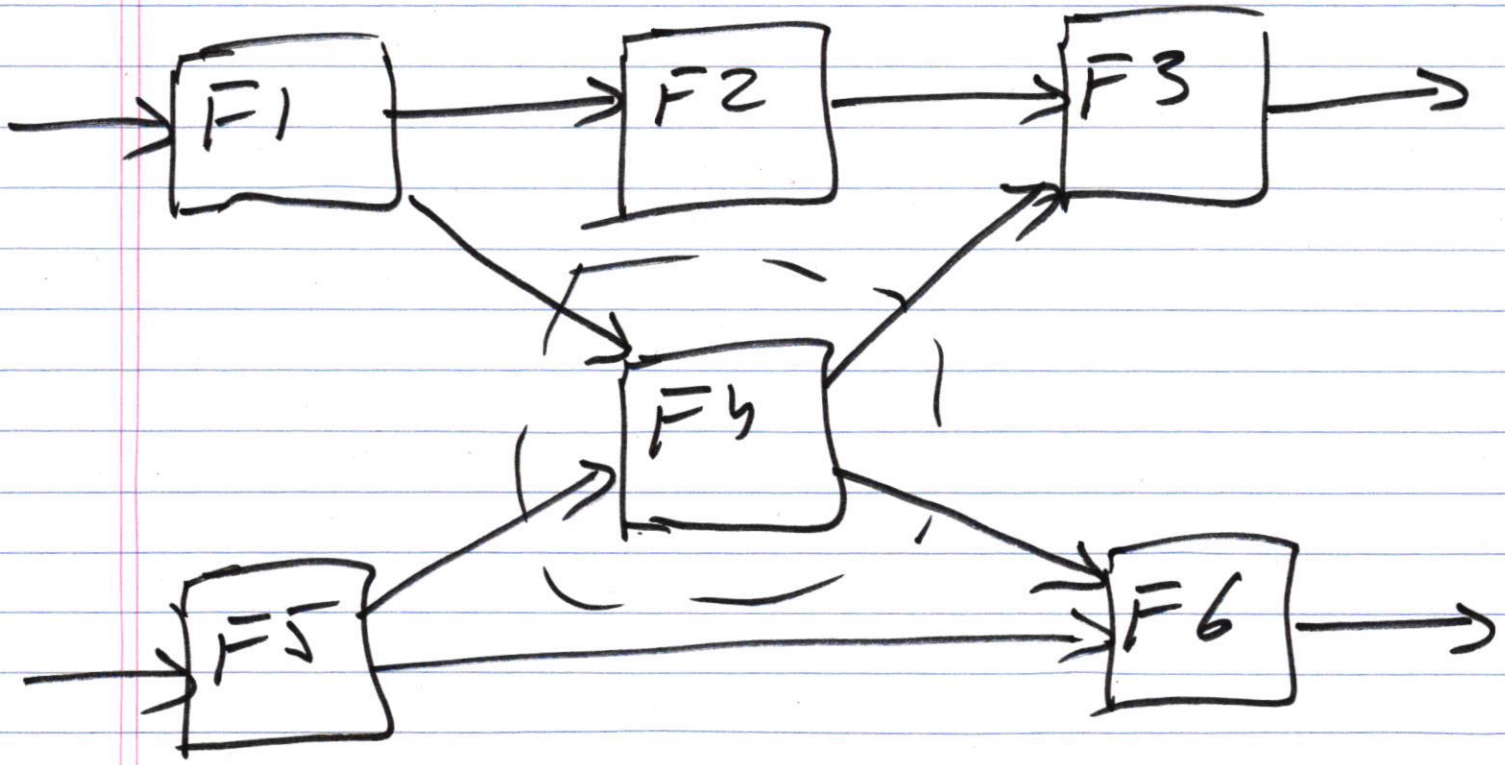


filter

data →

pipe


input pipes
output pipes
Filter

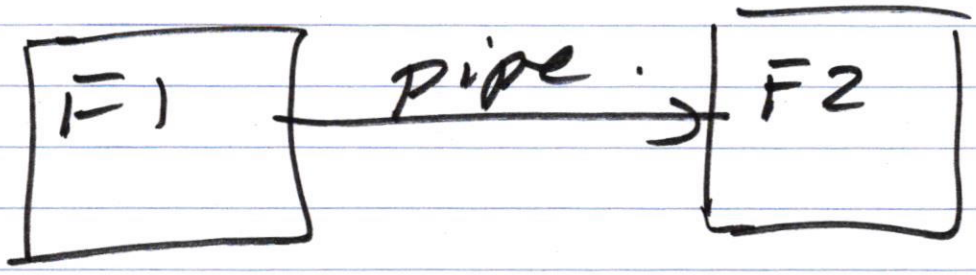

F1
F2
F3
F4
F5
F6

* Filters are independent entities

* No global data structures

* Filters are not aware of other filters in the system.

Design issues



Filters

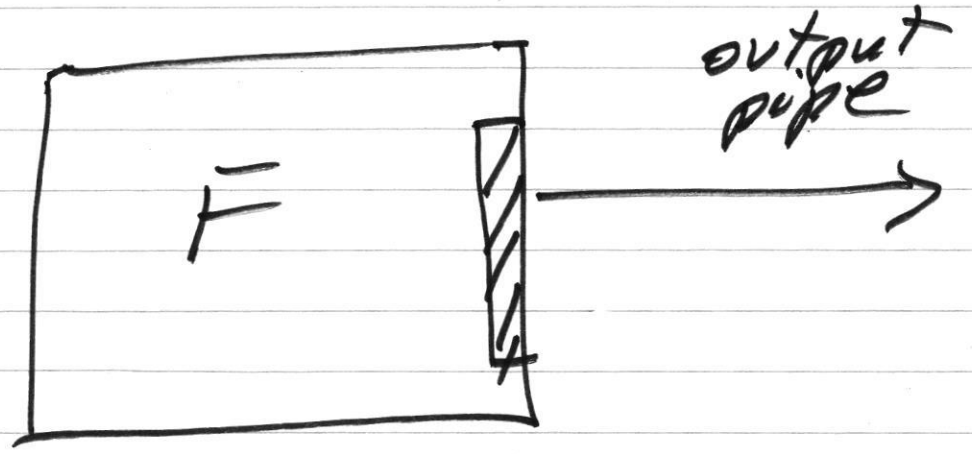
* passive filters
* active filters.

Pipes

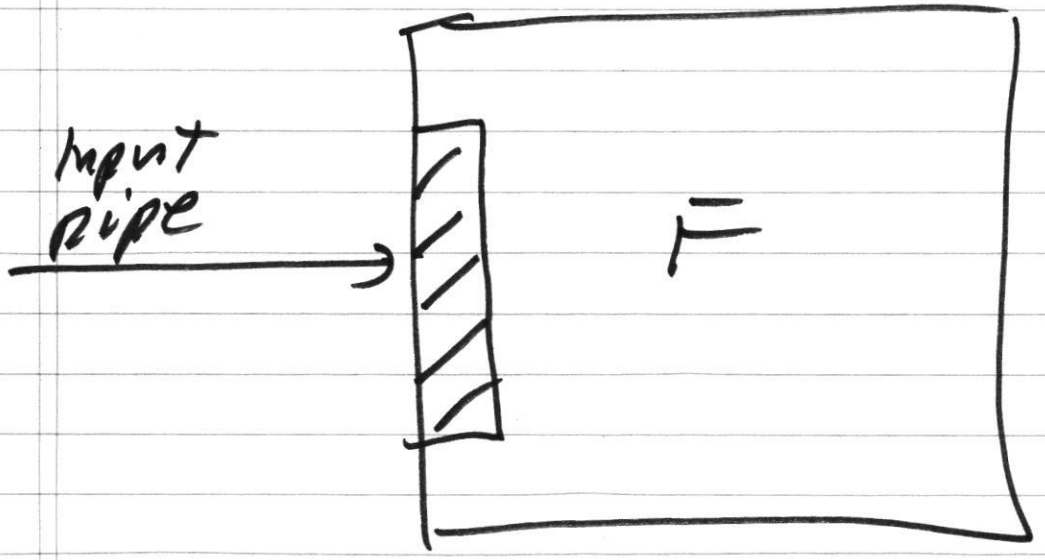
* unbuffered pipes
* buffered pipes

Passive filters

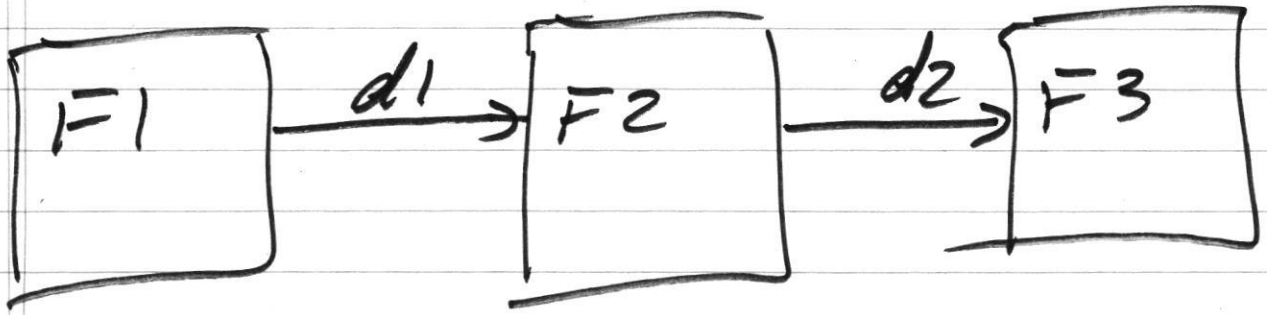
1. with pull-out pipes.

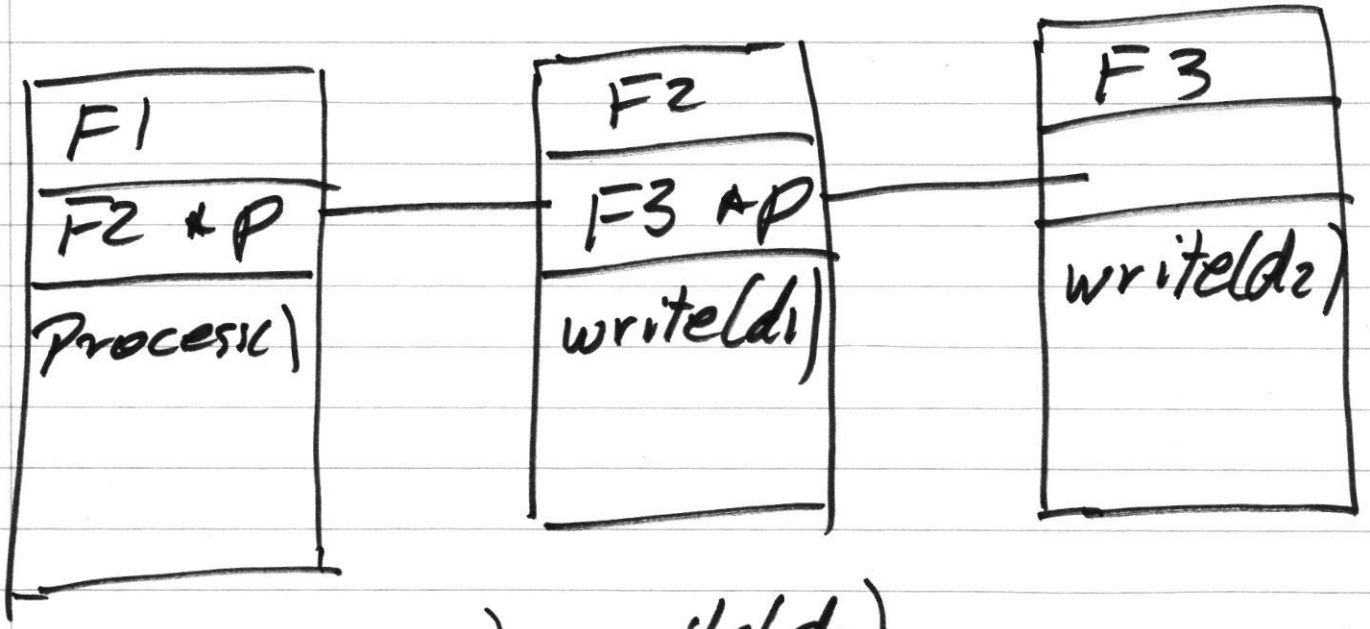


2. with push pipes



I. passive filters
unbuffered pipes
push pipes



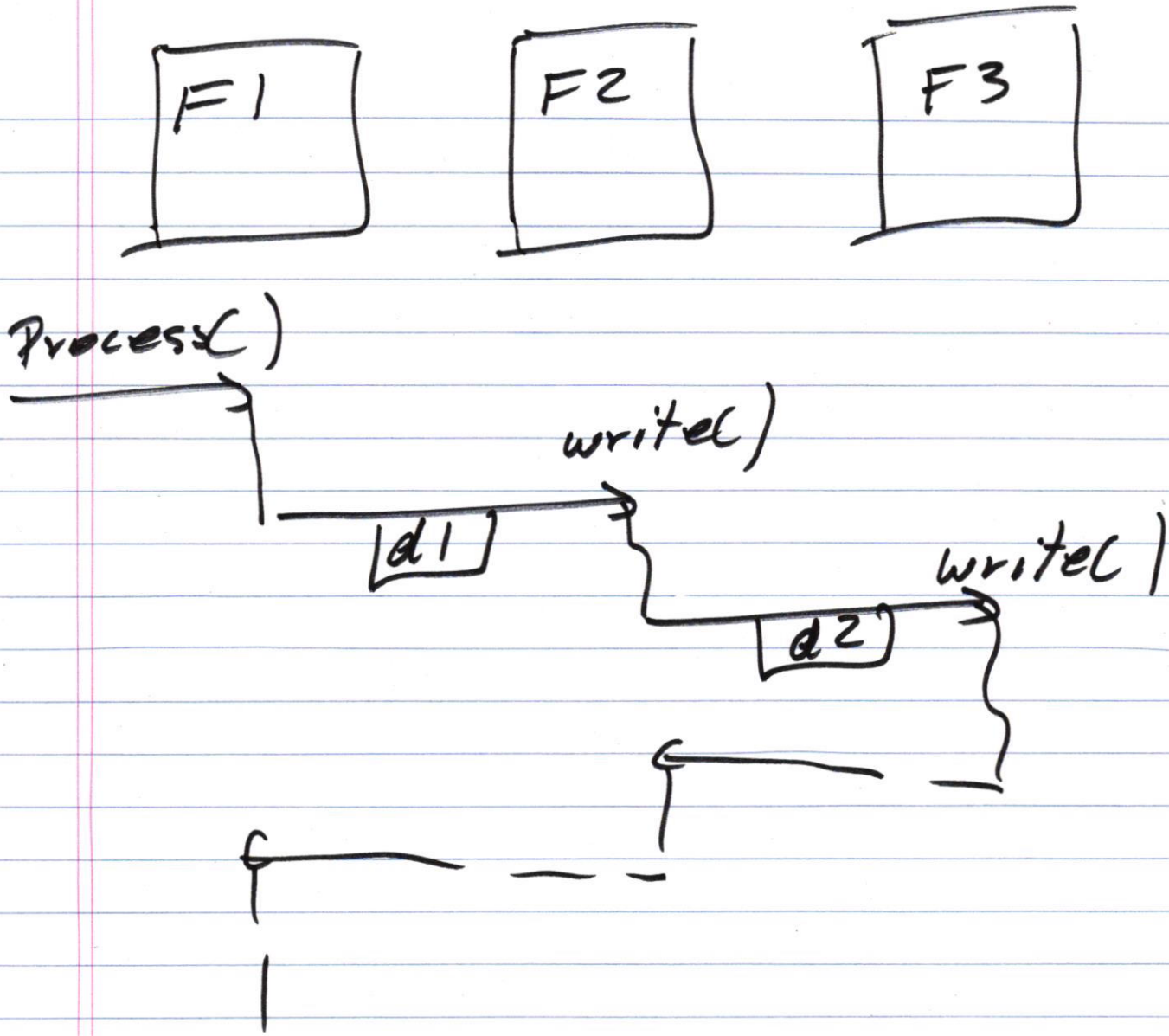


Process()

p->write(d1)

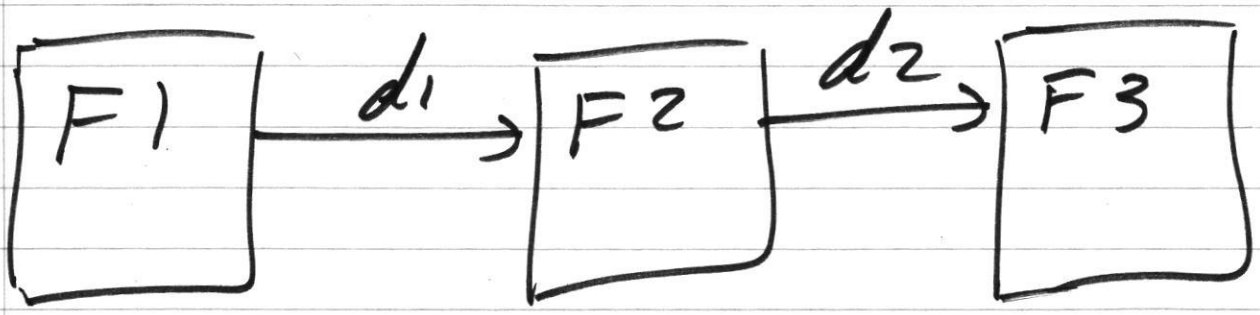
write(d1)

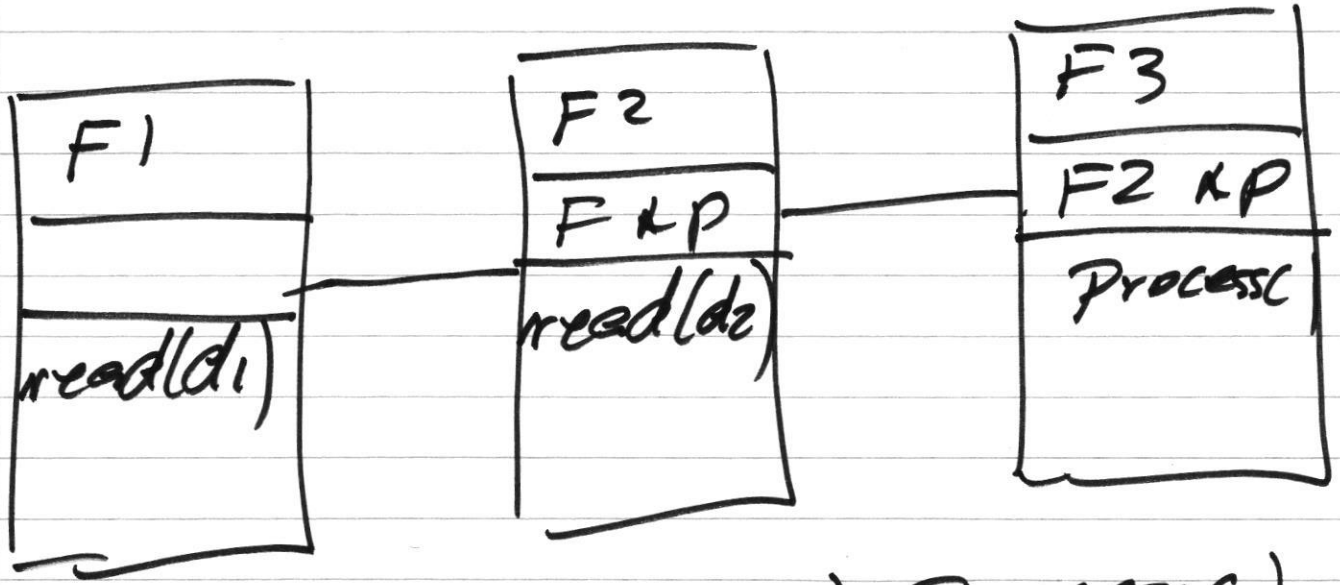
p->write(d2)


F1
F2
F3
Process()
write()
d1
write()
d2

II

* passive filters.
* unbuffered pipes.
* pull-out pipes





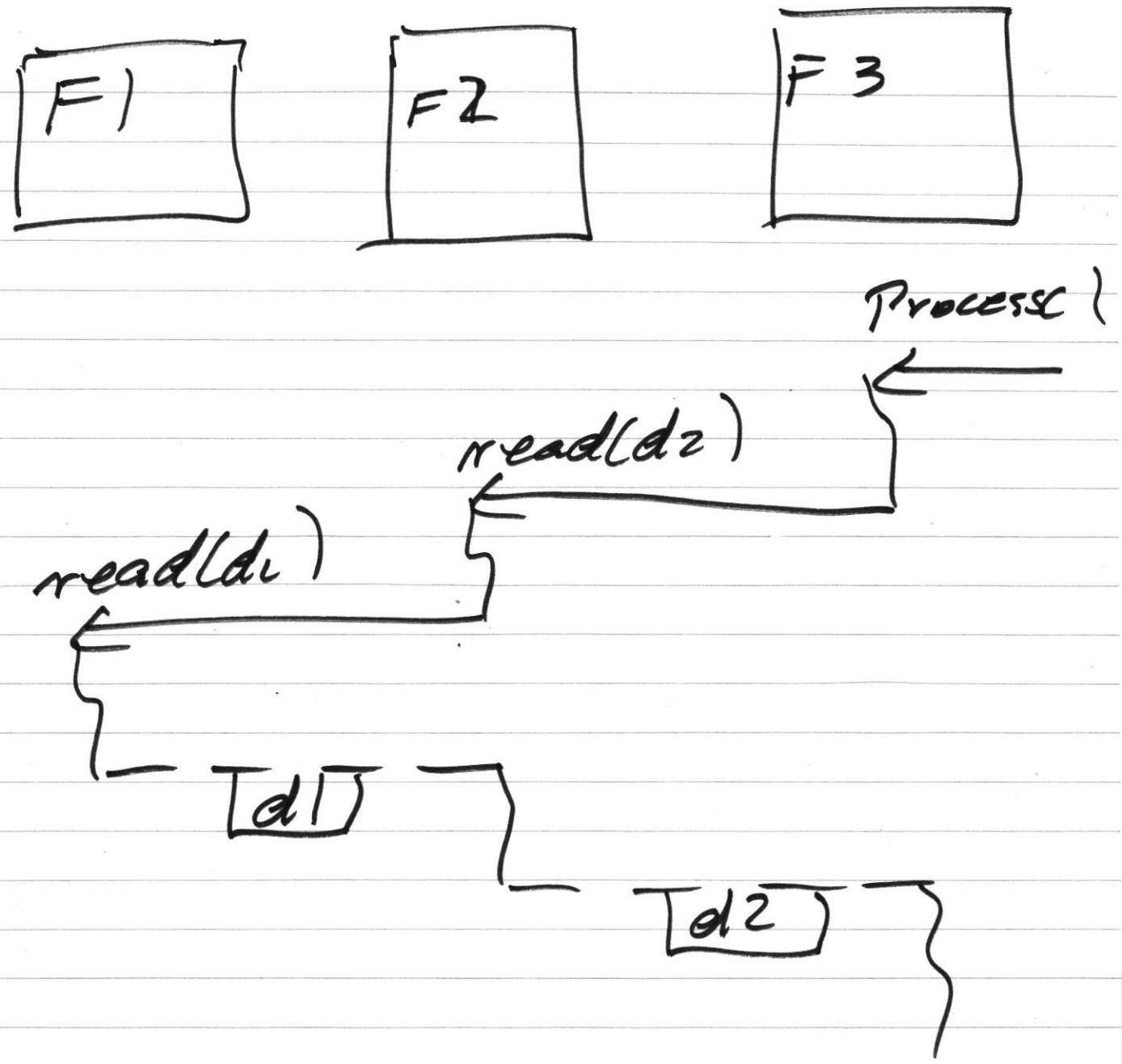
read(d2)

p->read(d1)

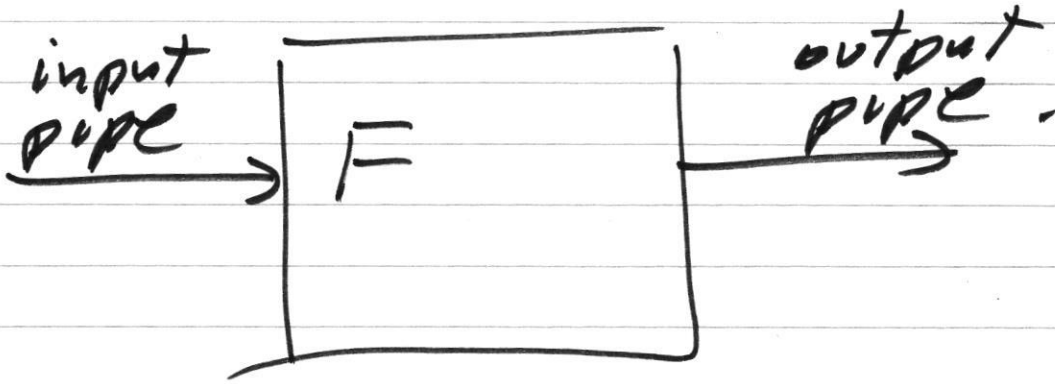
return d2

Process()

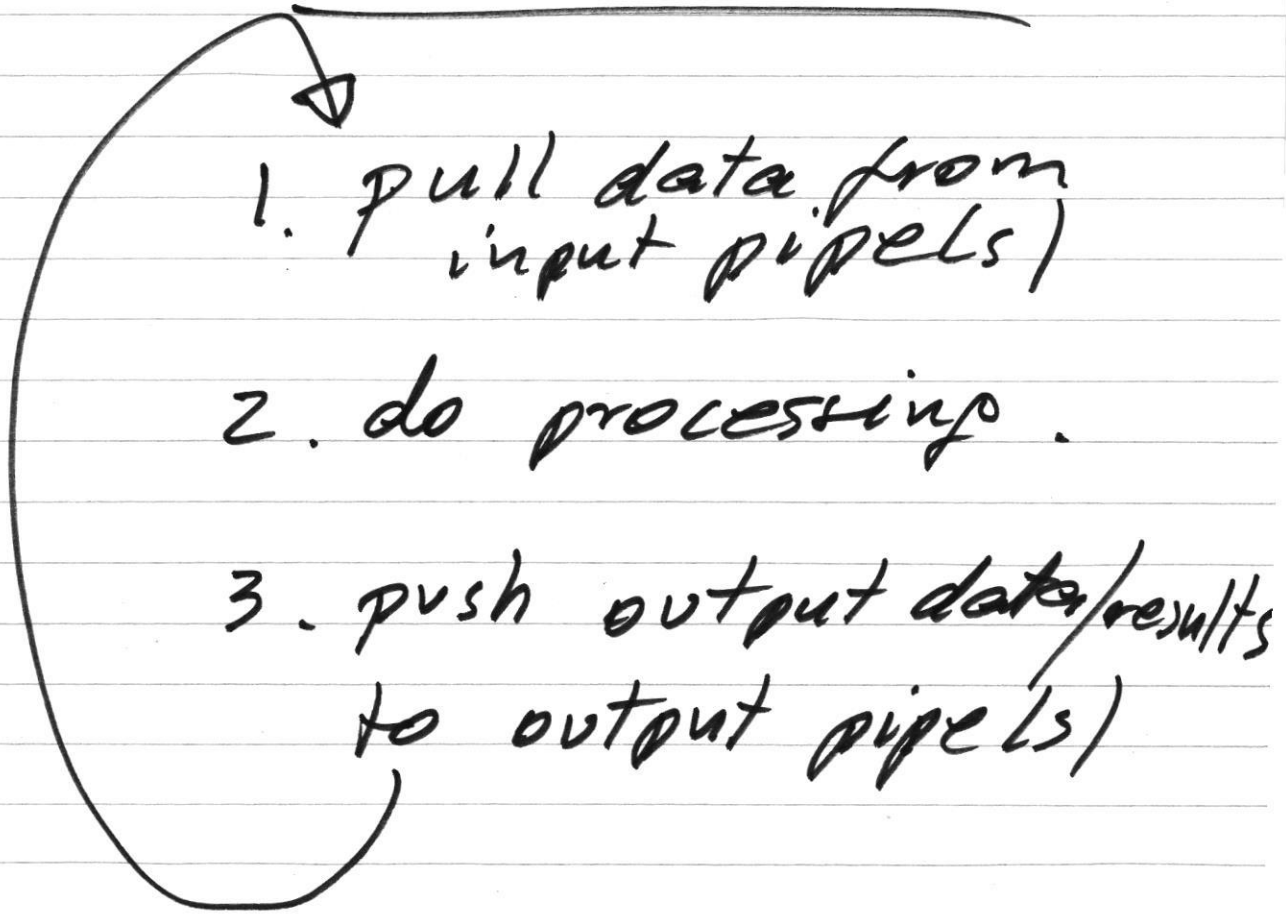
p->read(d2)


F1
F2
F3
Process
read(d2)
read(d1)
d1
d2

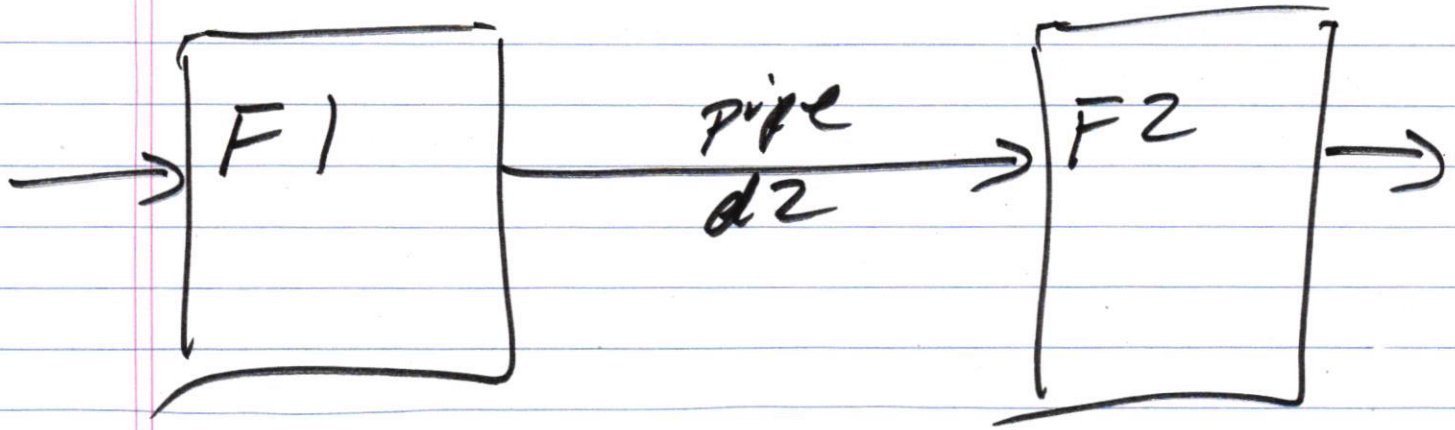
Active filters

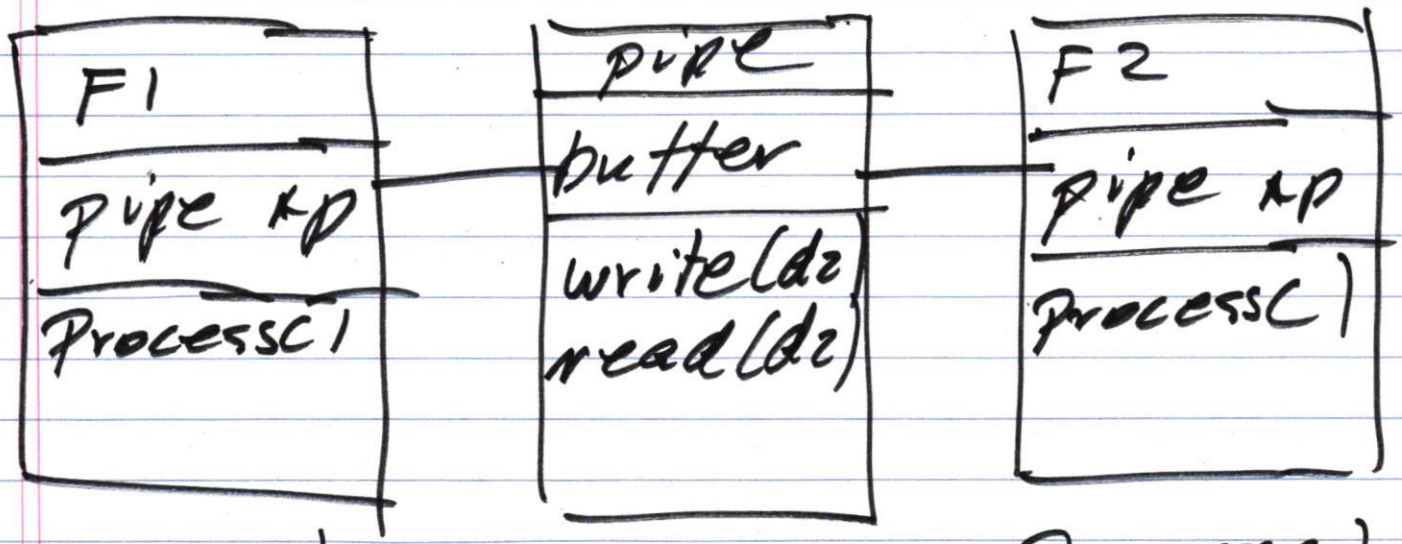


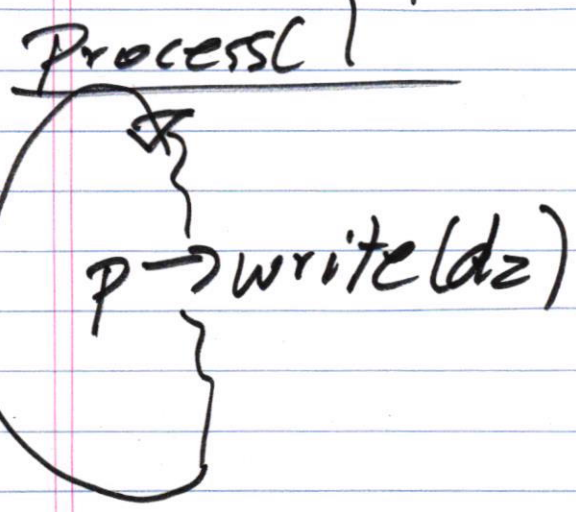
active loop

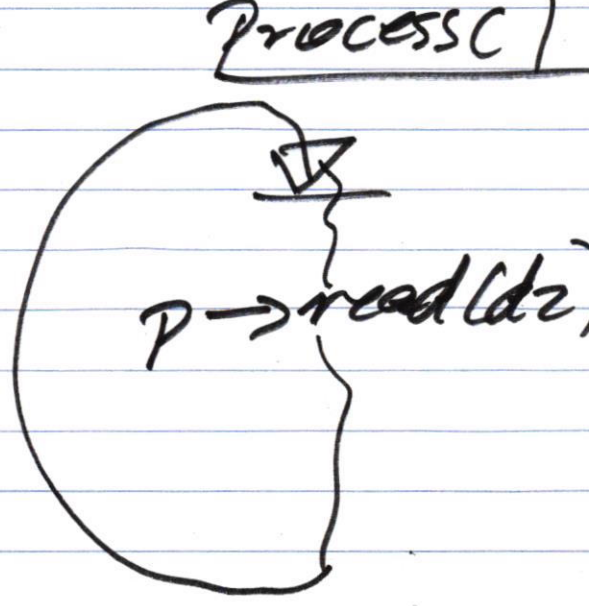


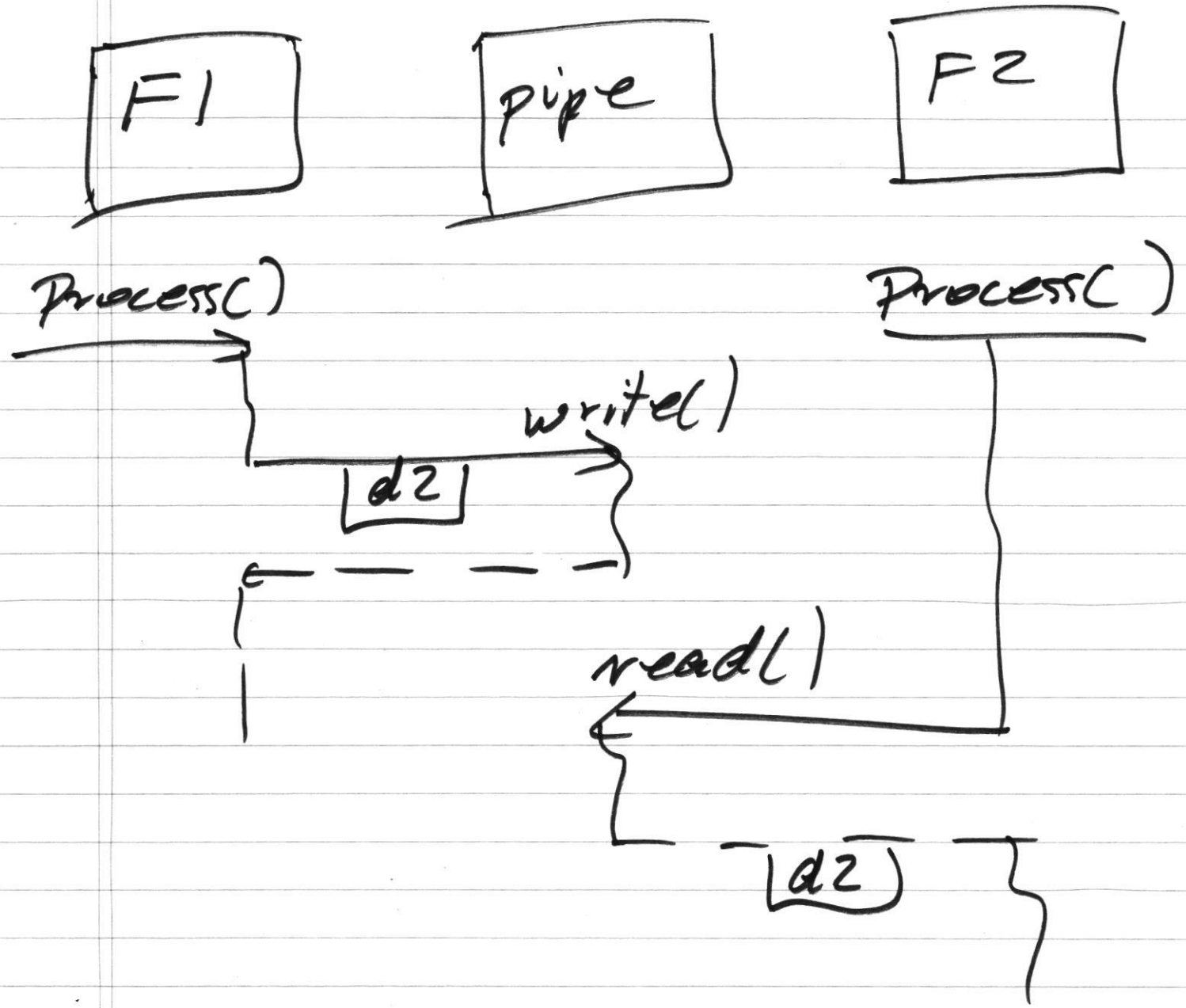
III * Active filters
* buffered pipes

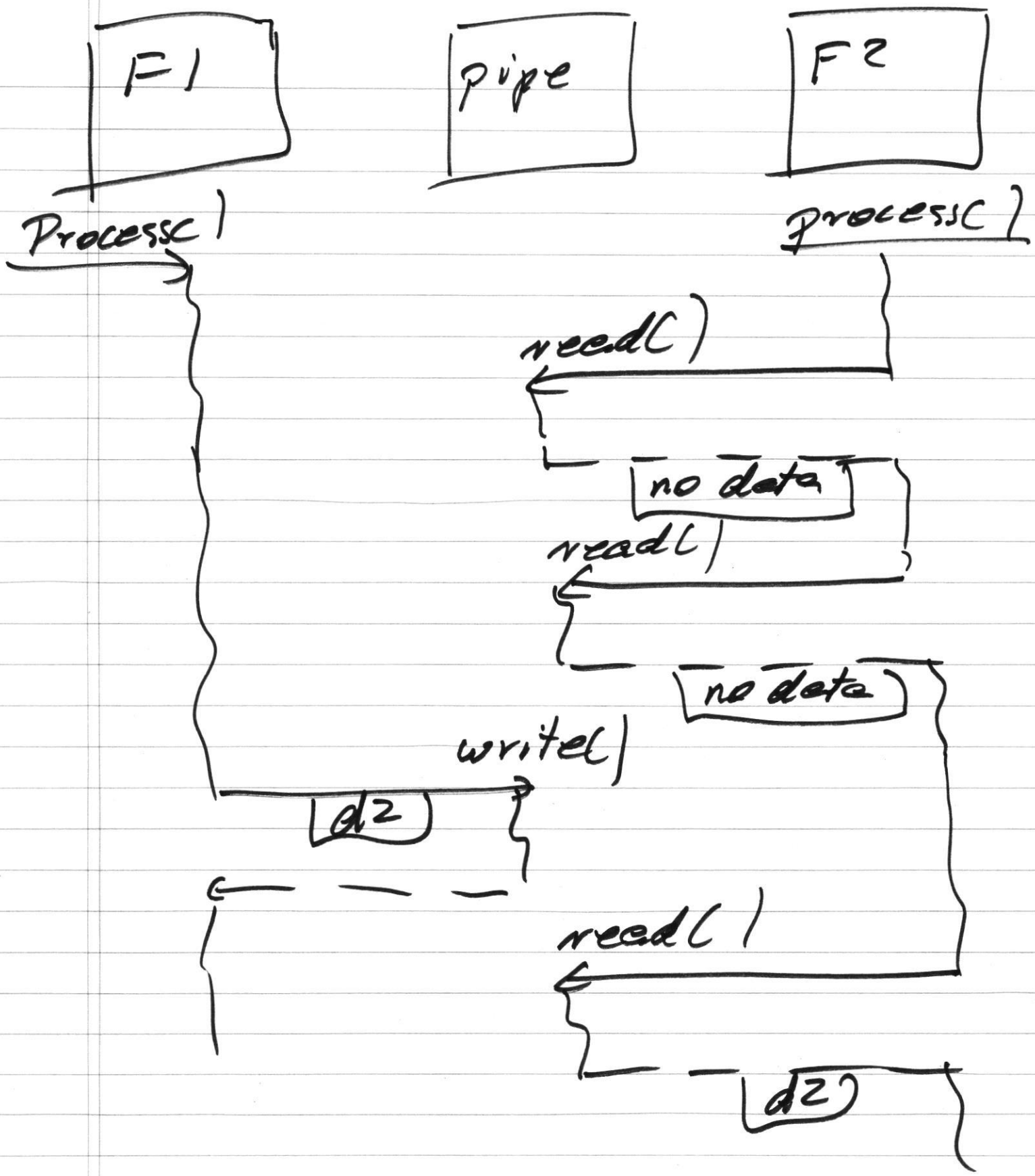


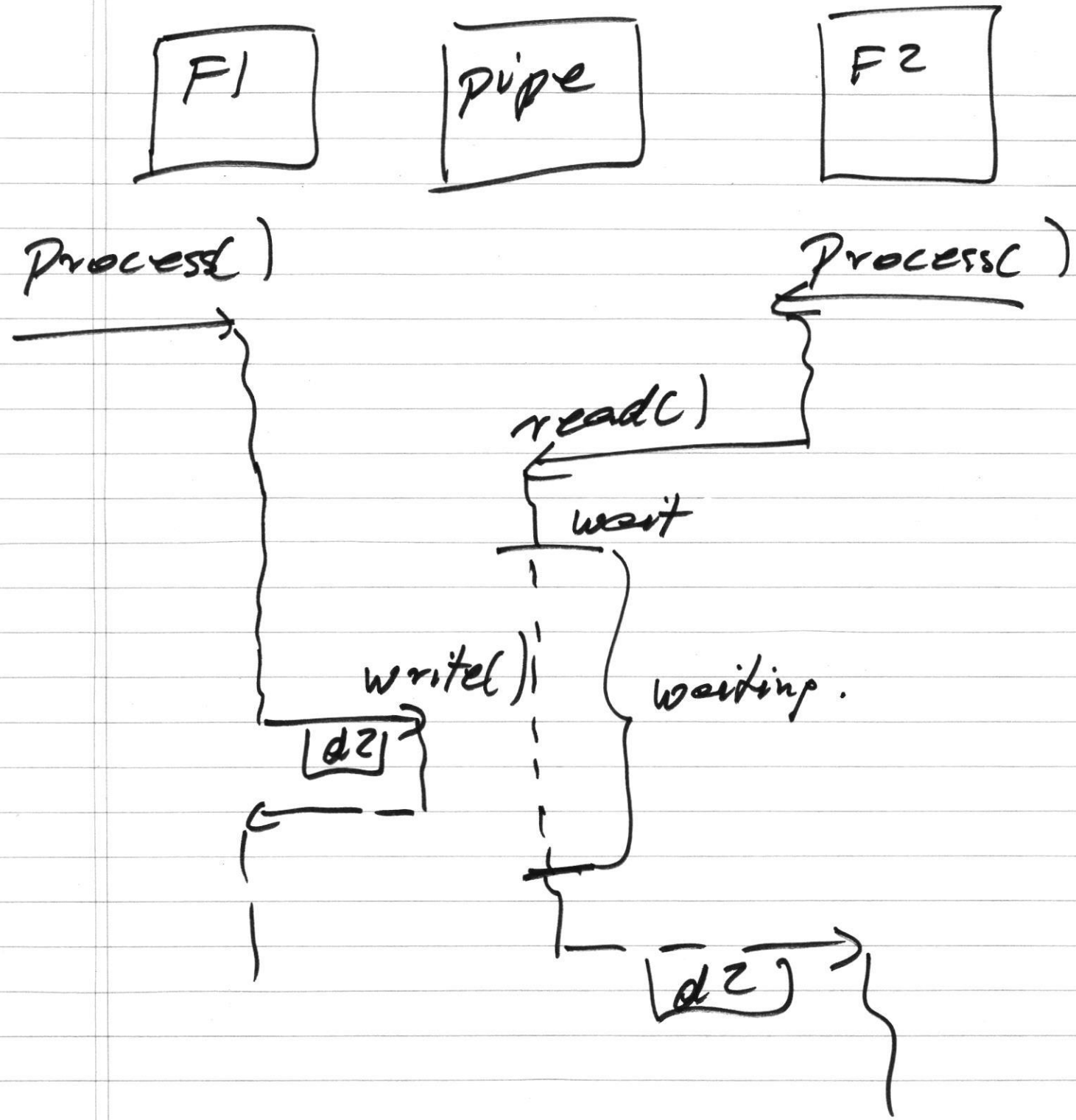




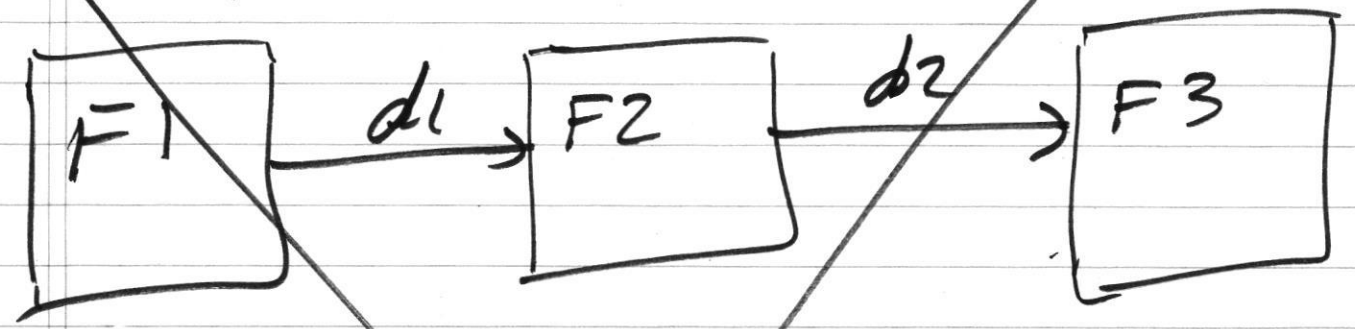


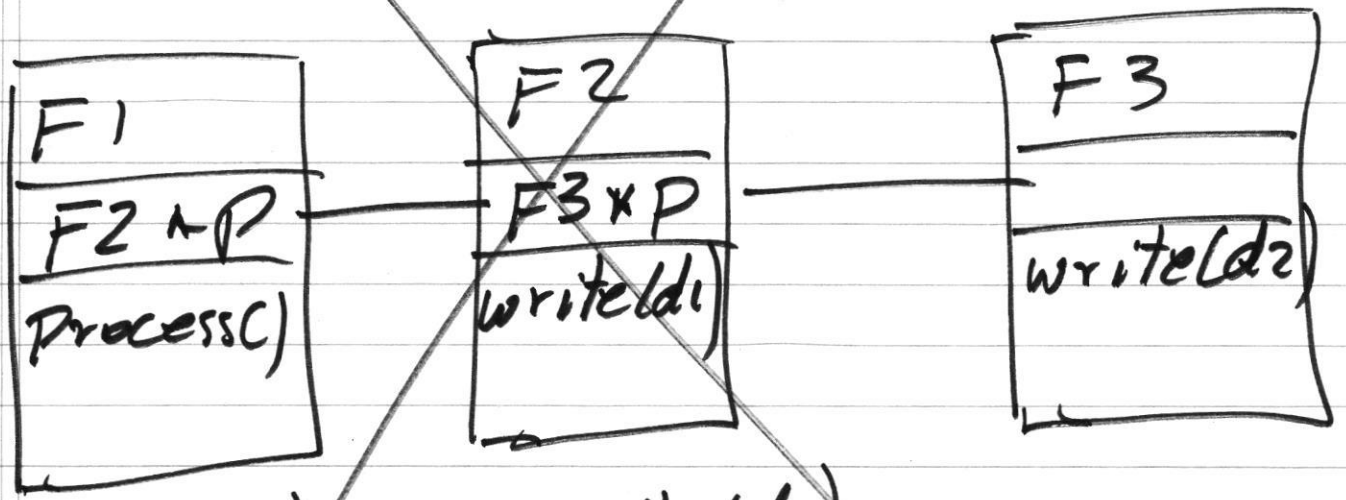

F1
pipe
F2
Process()
Process()
write()
d2
read()
d2

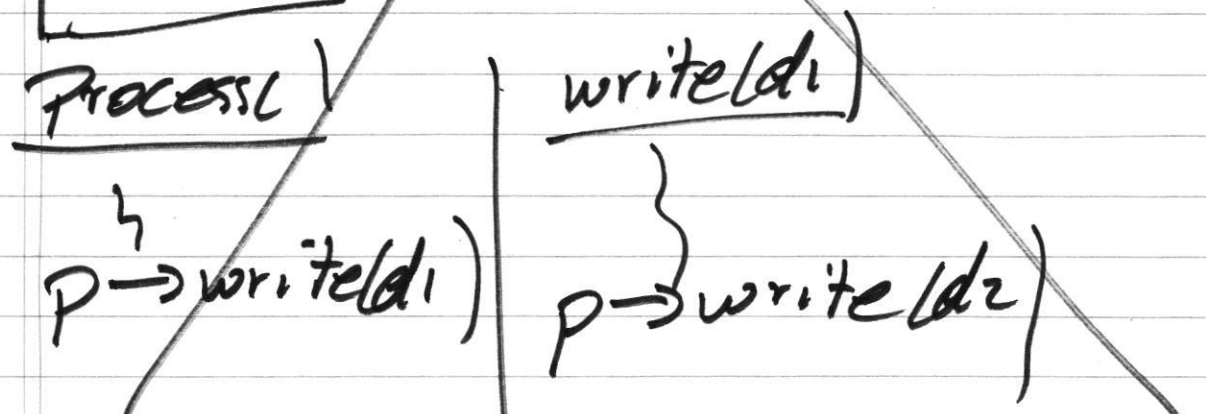

F1
pipe
F2
Process C1
Process C2
read()
no data
read()
no data
write()
d2
read()
d2

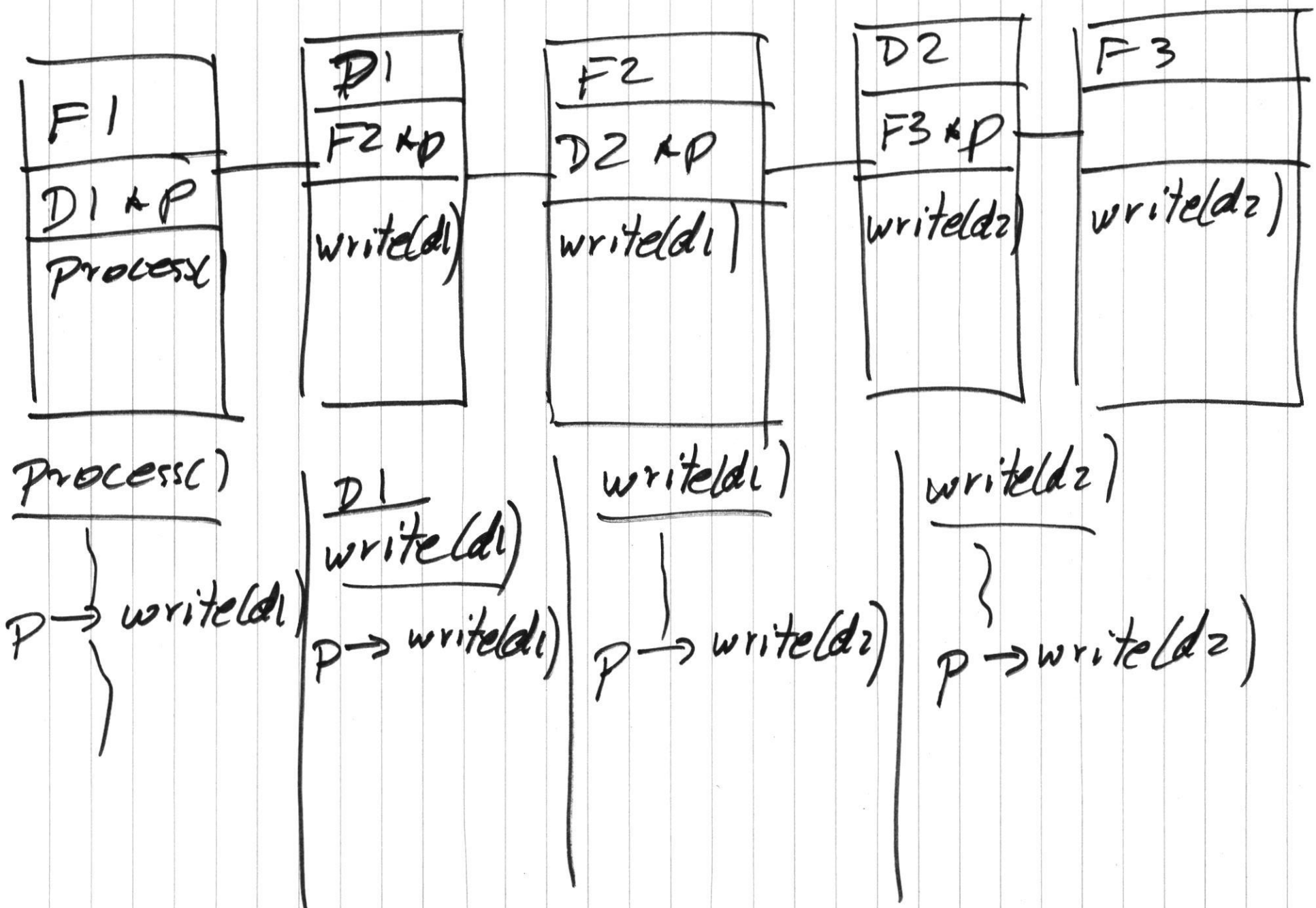

F1
pipe
F2
Process(
Process(
read()
wait
write()
waiting.
d2
d2

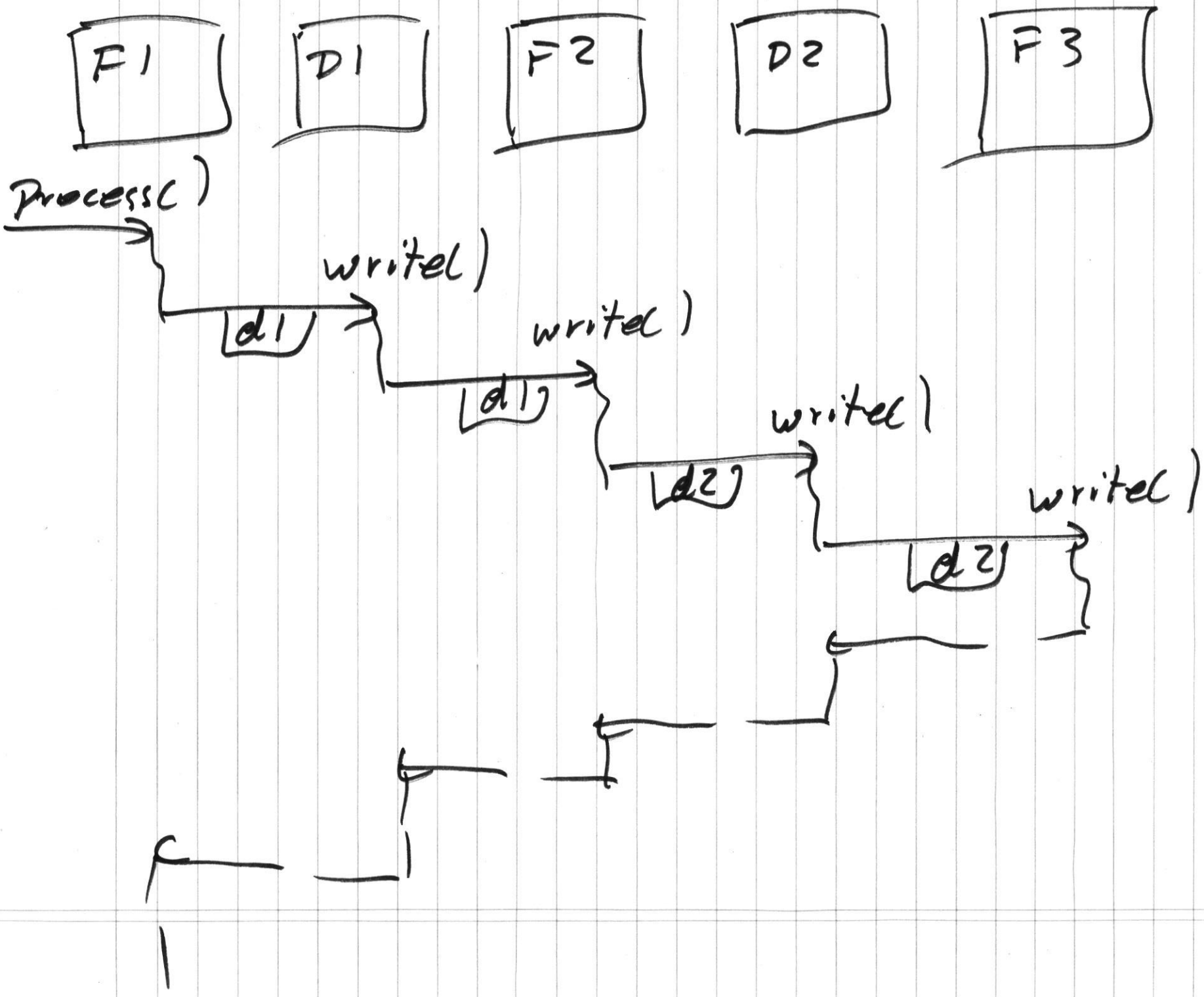
I passive filters
unbuffered pipes
push pipes.



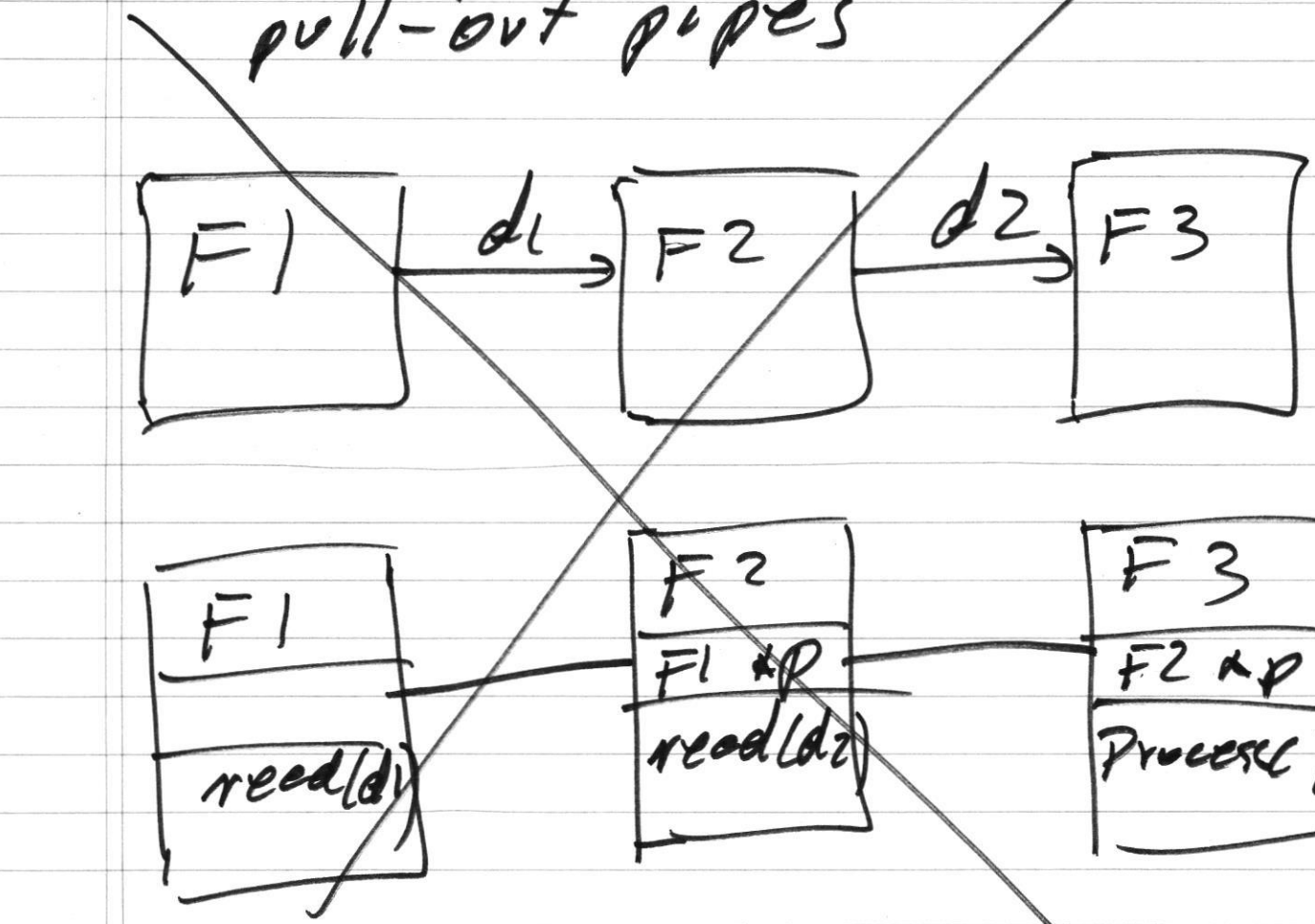


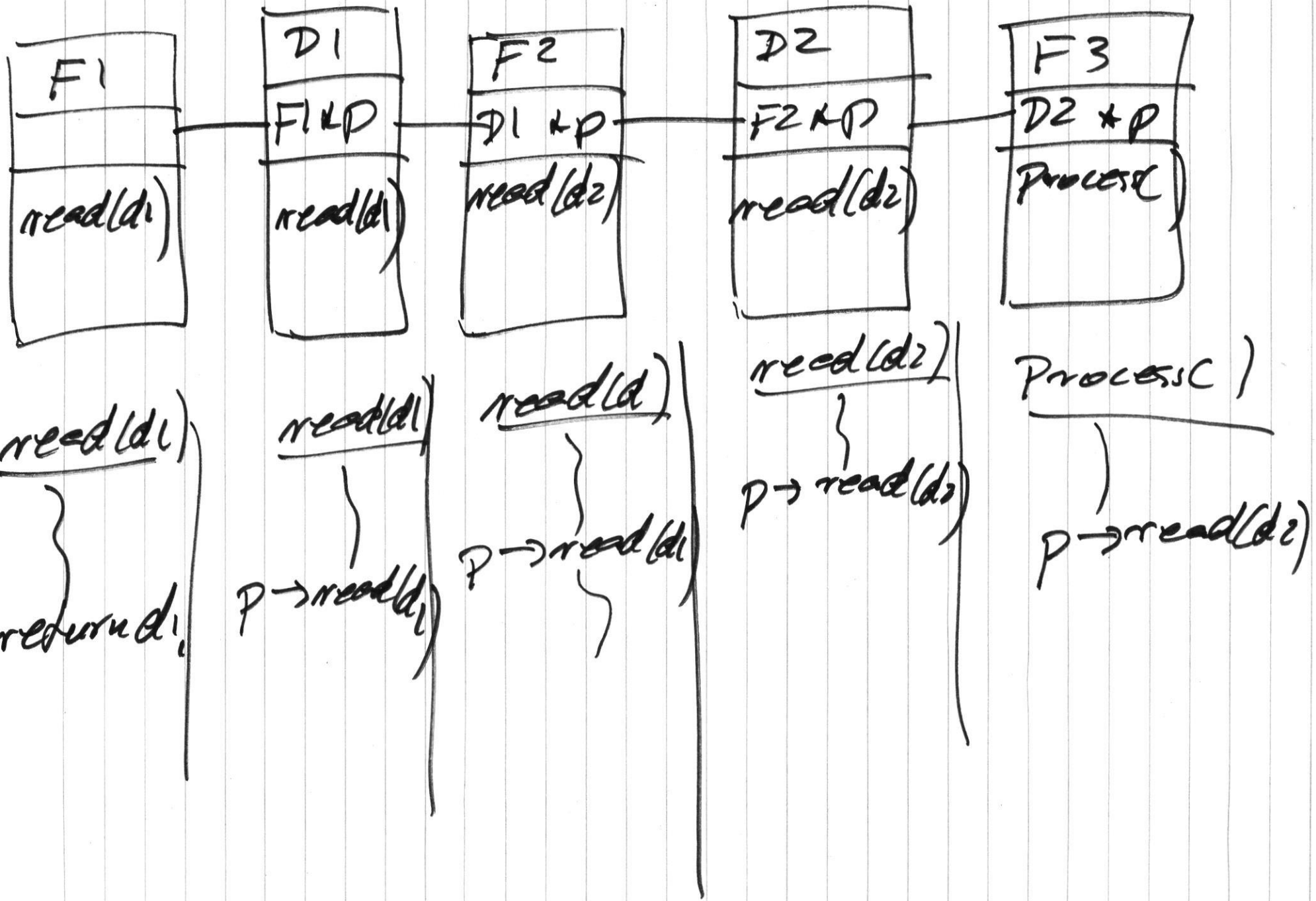


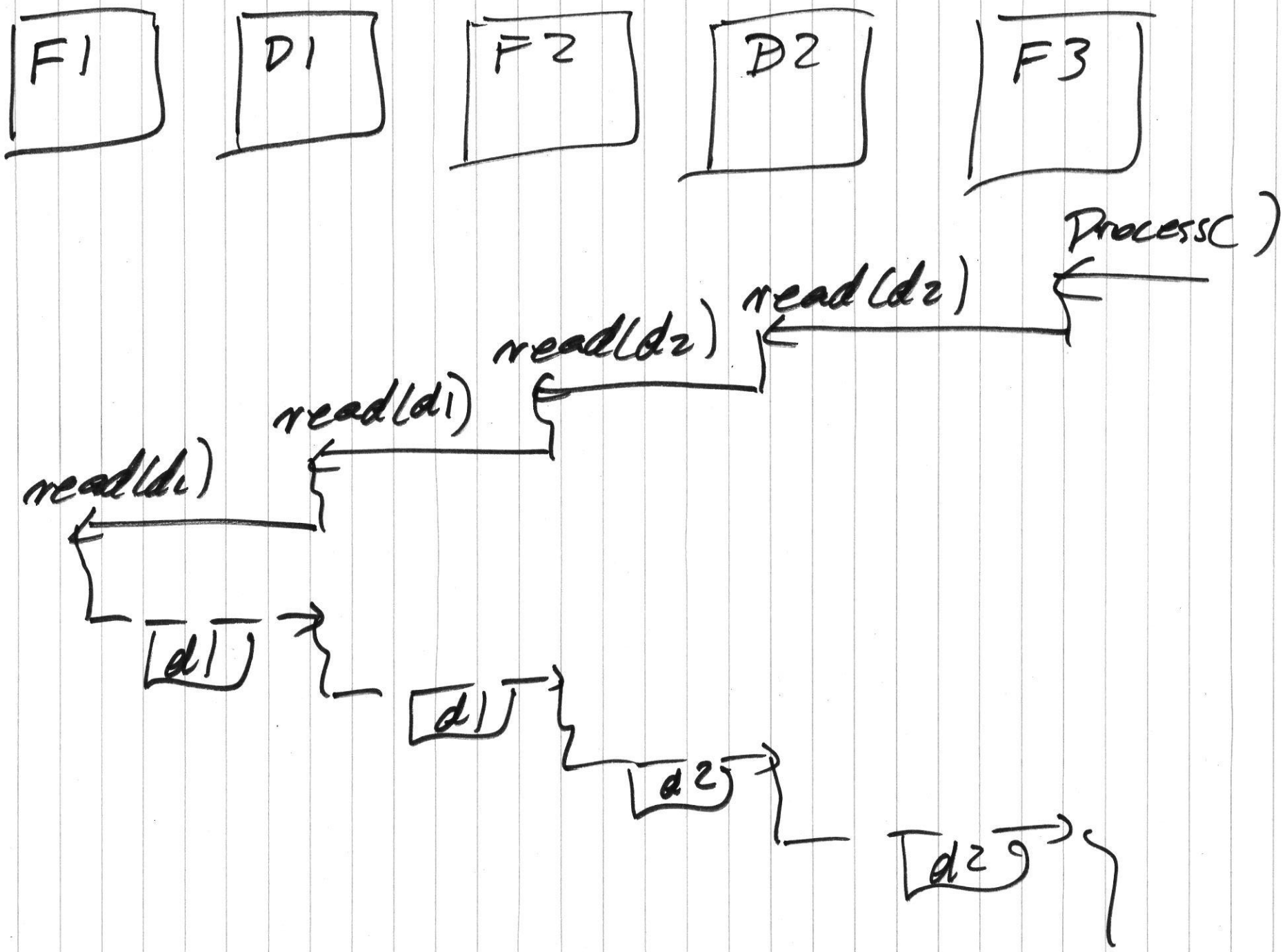

F1
D1 *p
Process()
D1
F2 *p
write(d1)
F2
D2 *p
write(d1)
D2
F3 *p
write(d2)
F3
write(d2)
Process()
p → write(d1)
D1
write(d1)
p → write(d1)
write(d1)
p → write(d2)
write(d2)
p → write(d2)


F1
D1
F2
D2
F3
Process()
write()
d1
write()
d1
write()
d2
write()
d2

II Passive filters
unbuffered pipes
pull-out pipes

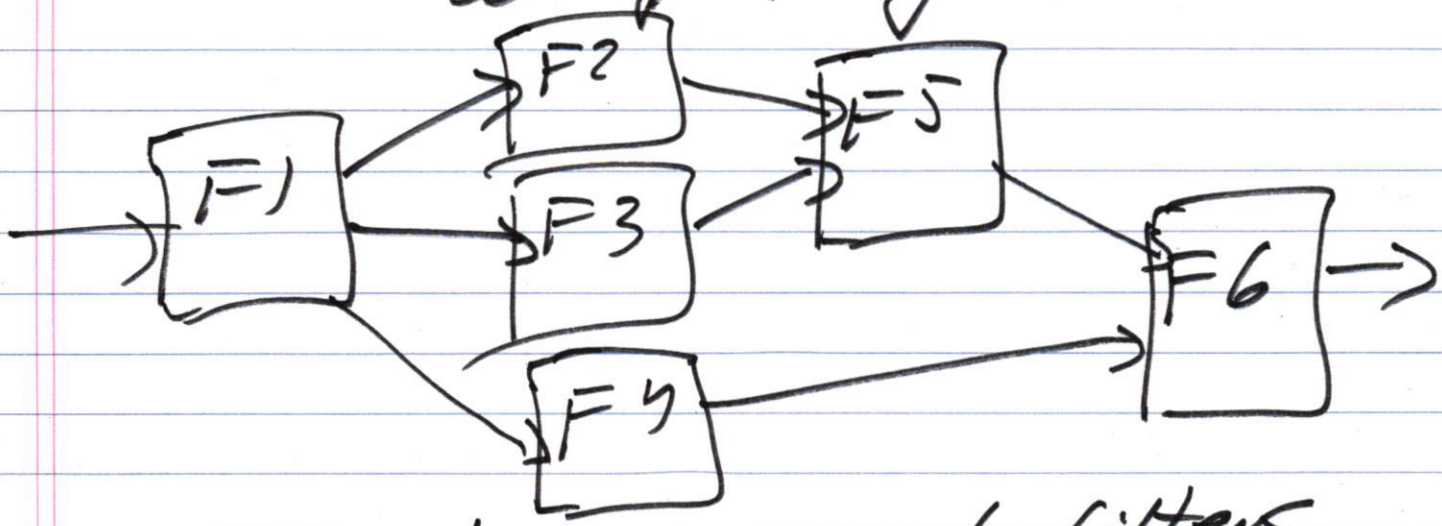



F1
read(d1)
D1
F1 *p
read(d1)
F2
D1 *p
read(d2)
D2
F2 *p
read(d2)
F3
D2 *p
Process()
read(d1)
return d1
read(d1)
p->read(d1)
read(d)
p->read(d1)
read(d2)
p->read(d2)
Process()
p->read(d2)


F1
D1
F2
D2
F3
Process()
read(d2)
read(d2)
read(d1)
read(d1)
d1
d1
d2
d2

Advantages

* it is easy to understand the overall architecture of the system
* easy to maintain.
  * easy to replace filters.
  * easy to add new filters.
* supports distributed computing.



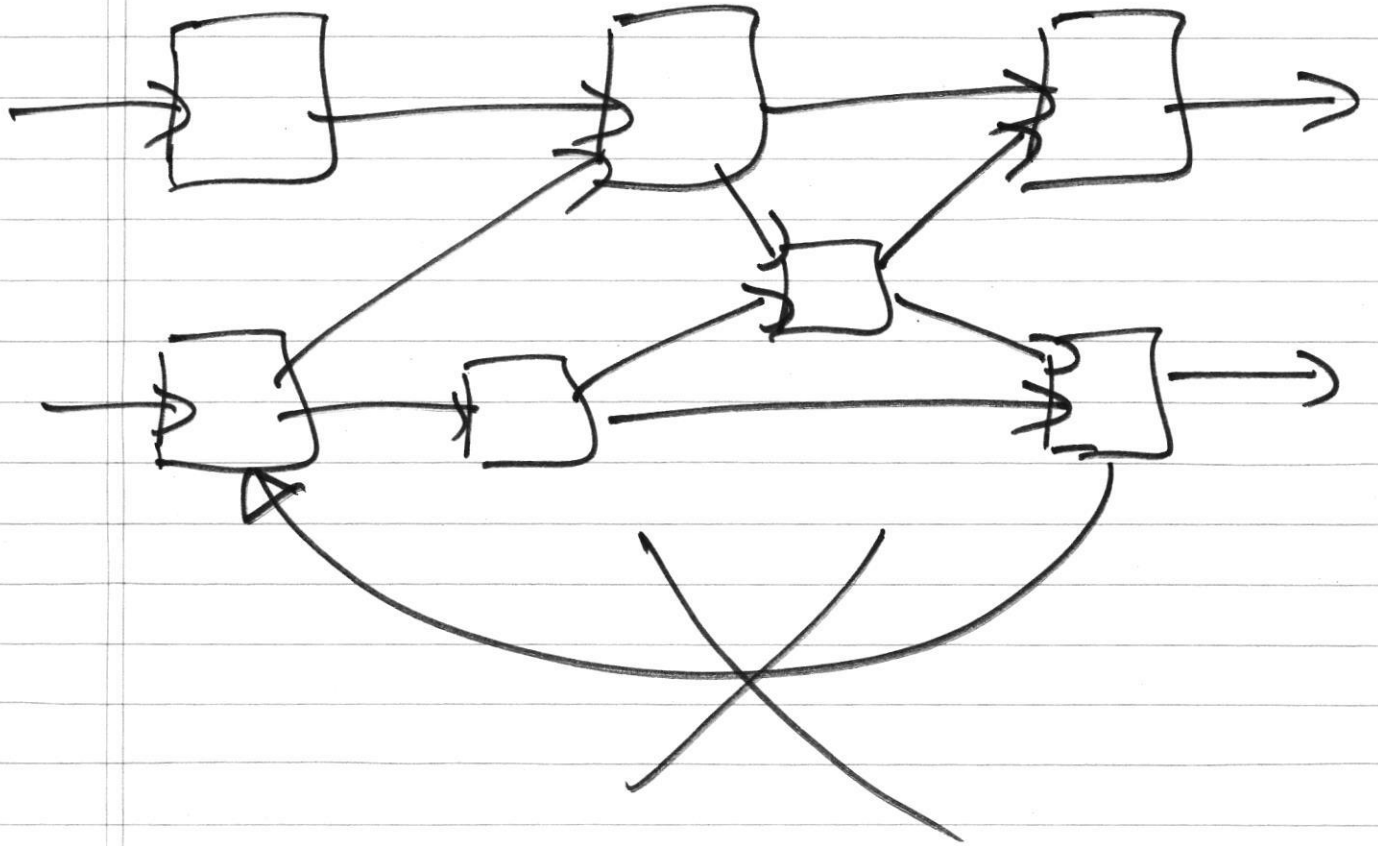
* supports reuse of filters.
  * library of filters

Disadvantages

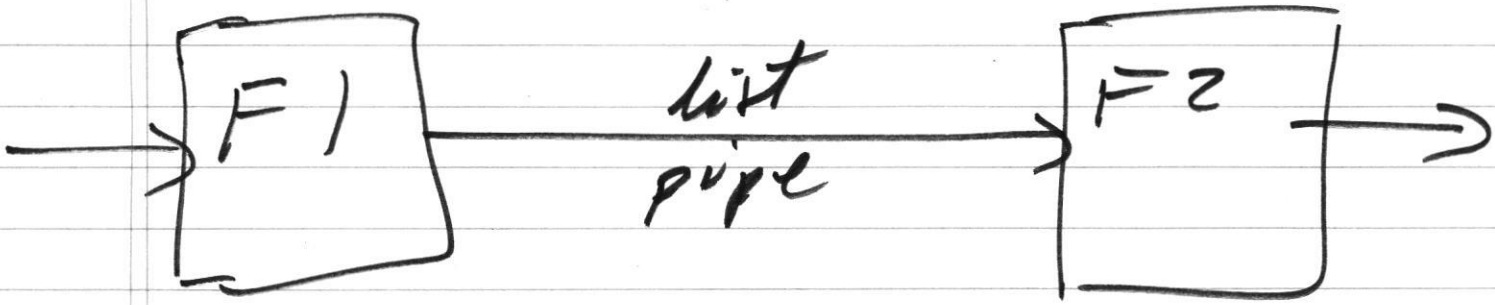
* not appropriate for some types of systems
  e.g. interactive systems

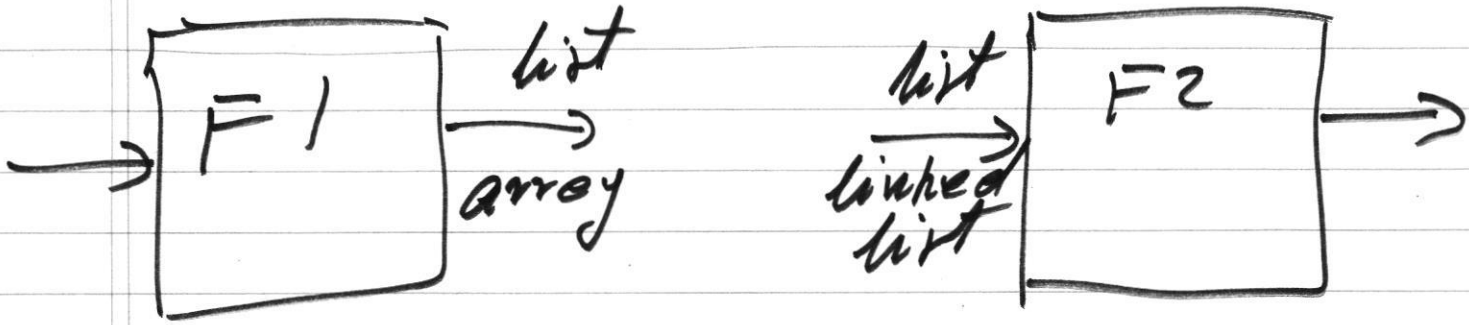
---

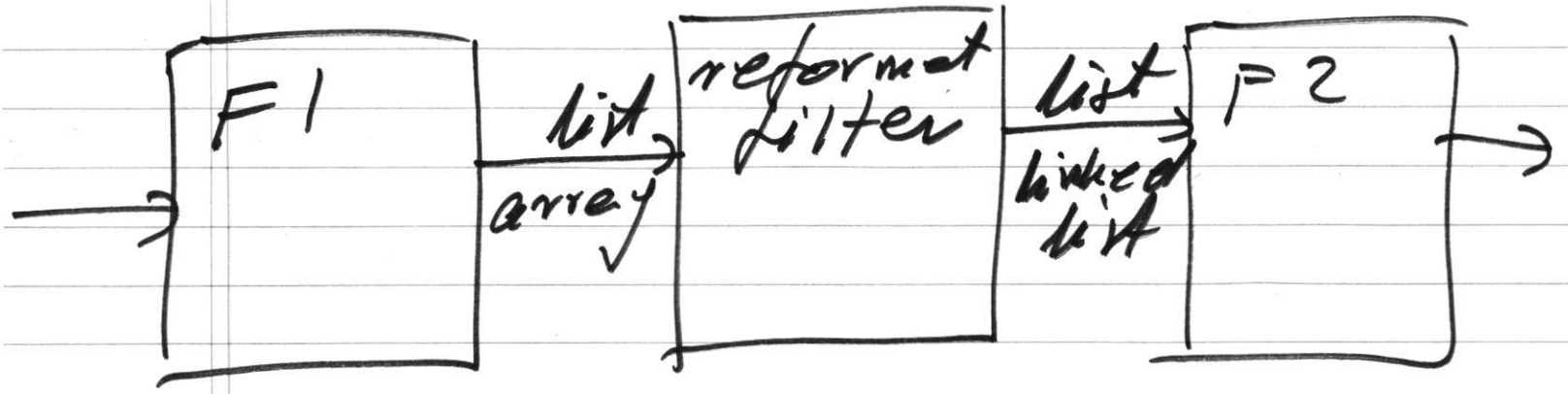
* architecture
  acyclic graph



* issues of matching input/output pipes.



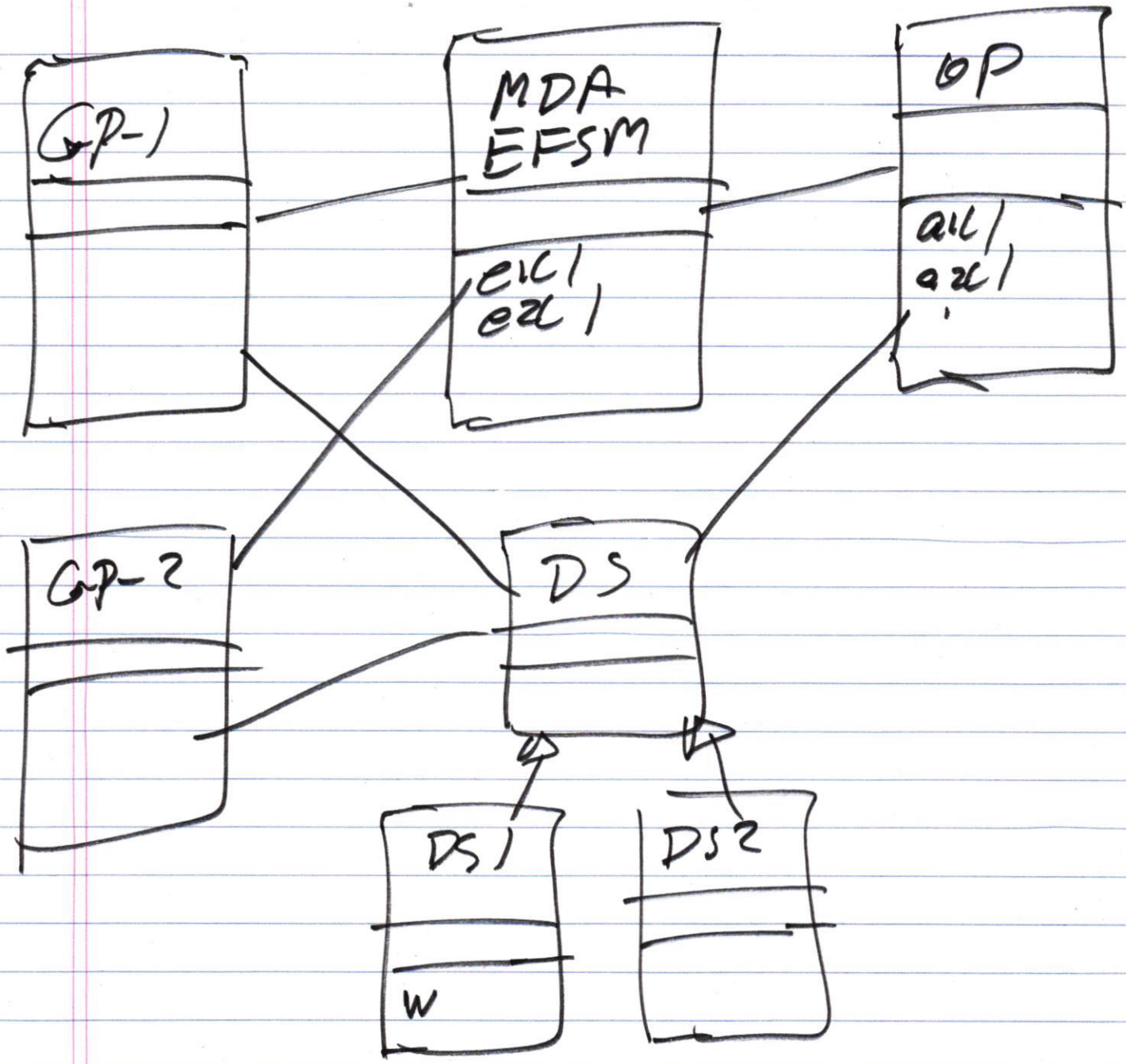




Project Part #1

MDA-EFSM

Deadline: Monday, April 1


GP-1
MDA EFSM
e1c1, e2c1
OP
a1c1, a2c1
GP-2
DS
DS1
w
DS2

meta events	Not meta events
Activate()	Diesel, Regular, Premium — No
start()	
PayCredit()	
Reject()	
Cancel()	Gallon, Liter — No!!
Approved	
PayCash()	
startPump()	PumpGallon — No
SelectGrade()	
Pump()	

PROJECT-General Description
PART #1

CS 586; Spring 2024
Deadlines:
Part #1: MDA-EFSM (5 points): Monday, April 1, 2024
Late submissions: 50% off
After **April 4** the MDA-EFSM will not be accepted.

This is an **individual** project, not a team project.

Submission: The MDA-EFSM assignment must be submitted on Blackboard. Your submission should be as a **single PDF-file** (otherwise, a 10% penalty will be applied). The hardcopy submissions will not be accepted.

The detailed description of Part #2 of the project will be posted later.

Goal:
The goal of this project is to design two different gas pump components using the Model-Driven Architecture (MDA) and then implement these gas pump components based on this design.

Description of the Project:
There are two gas pump components: *GP-1* and *GP-2*.

The gas pump **GP-1** component supports the following operations:

Activate (int a)	// the gas pump is activated where *a* is the price of the gas per liter
Start()	//start the transaction
PayCredit()	// pay for gas by a credit card
Reject()	// credit card is rejected
Cancel()	// cancel the transaction
Approved()	// credit card is approved
PayCash(int c)	// pay for gas by cash, where *c* represents prepaid cash
StartPump()	// start pumping gas
Pump()	// one liter of gas is disposed
StopPump()	// stop pumping gas

The **GasPump-2** component supports the following operations:

Activate (float a, float b, float c)	// the gas pump is activated where *a* is the price of Regular gas, *b* is //the price of Premium gas and *c* is the price of Diesel per Gallon
Start()	//start the transaction
PayCash(int c)	// pay for gas by cash, where *c* represents prepaid cash
Cancel()	// cancel the transaction
Premium()	// Premium gas is selected
Regular()	// Regular gas is selected
Diesel()	// Diesel gas is selected
StartPump()	// start pumping gas
PumpGallon()	// one Gallon of gas is disposed
Stop()	// stop pumping gas
Receipt()	// Receipt is requested
NoReceipt()	// No receipt

Both gas pumps are state-based components and are used to control simple gas pumps. Users can pay by cash or a credit card. The gas pump may dispose of different types of gasoline. The price of the gasoline is provided when the gas pump is activated. The detailed behavior of gas pump components is specified using EFSM. The EFSM of Figure 1 shows the detailed behavior of gas pump *GP-1* and the EFSM of Figure 2 shows the detailed behavior of gas pump *GP-2*. Notice that there are several differences between gas pump components.

Aspects that vary between two gas pump components:

a. Types of gasoline disposed
b. Types of payment
c. Display menu(s)
d. Messages
e. Receipts
f. Operation names and signatures
g. Data types
h. etc.

The goal of this project is to design two GP components using the Model-Driven Architecture (MDA) covered in the course. An executable meta-model referred to as MDA-EFSM of GP components should capture the "generic behavior" of two GP components and should be de-coupled from data and implementation details. Notice that in your design there should be **ONLY** one MDA-EFSM for two GP components. The meta-model (MDA-EFSM) used in the Model-Driven architecture should be expressed as an EFSM (Extended Finite State Machine) model. Notice that the EFSMs shown in Figure 1 and Figure 2 are **not acceptable** as a meta-model (MDA-EFSM) for this model-driven architecture.

MDA-ESM REPORT SUBMISSION

The MDA-EFSM model report for the *GP* components should contain:

- A class diagram
- A list of meta events for the MDA-EFSM
- A list of meta actions for the MDA-EFSM, where the responsibility of each action must be described
- A state diagram/model of the MDA-EFSM
- Pseudo-code of all operations of Input Processors of *GP-1* and *GP-2*

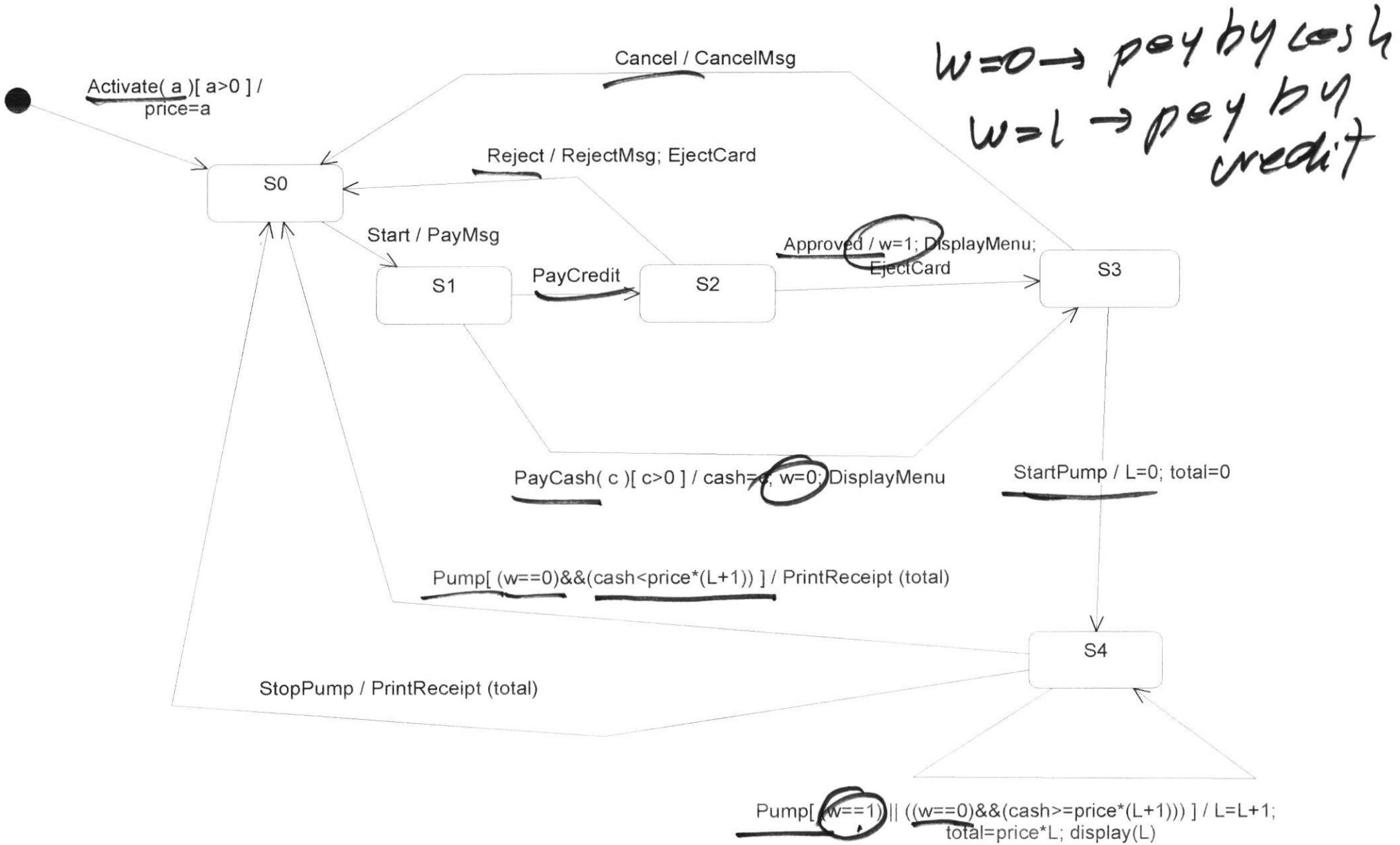


Figure 1: EFSM for gas pump GP-1

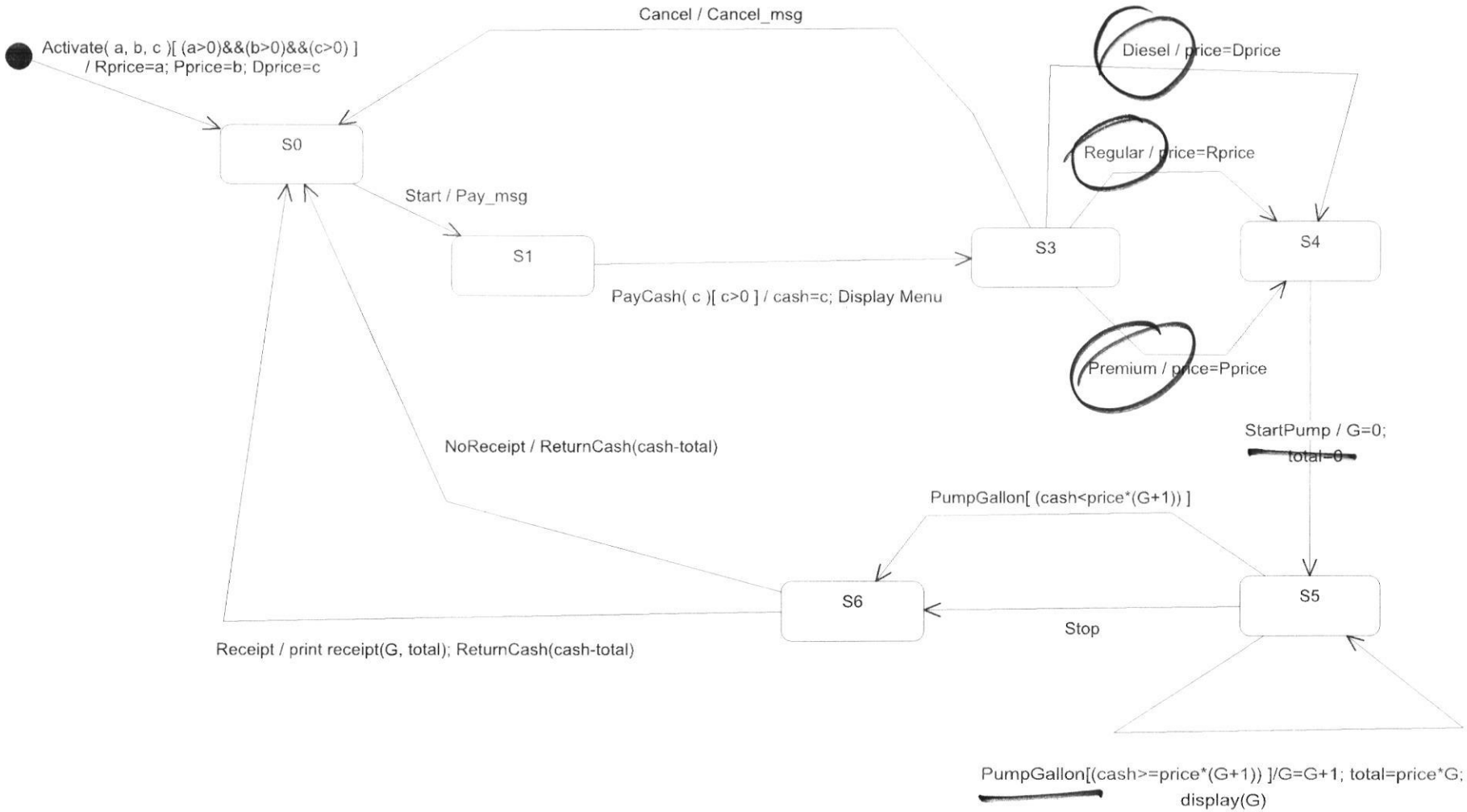


Figure 2: EFSM for GasPump-2