10/17/2022

# Operating System Project 2

synchronous inter-process communication through the use of both fork(), pipe(), and exec() system calls.

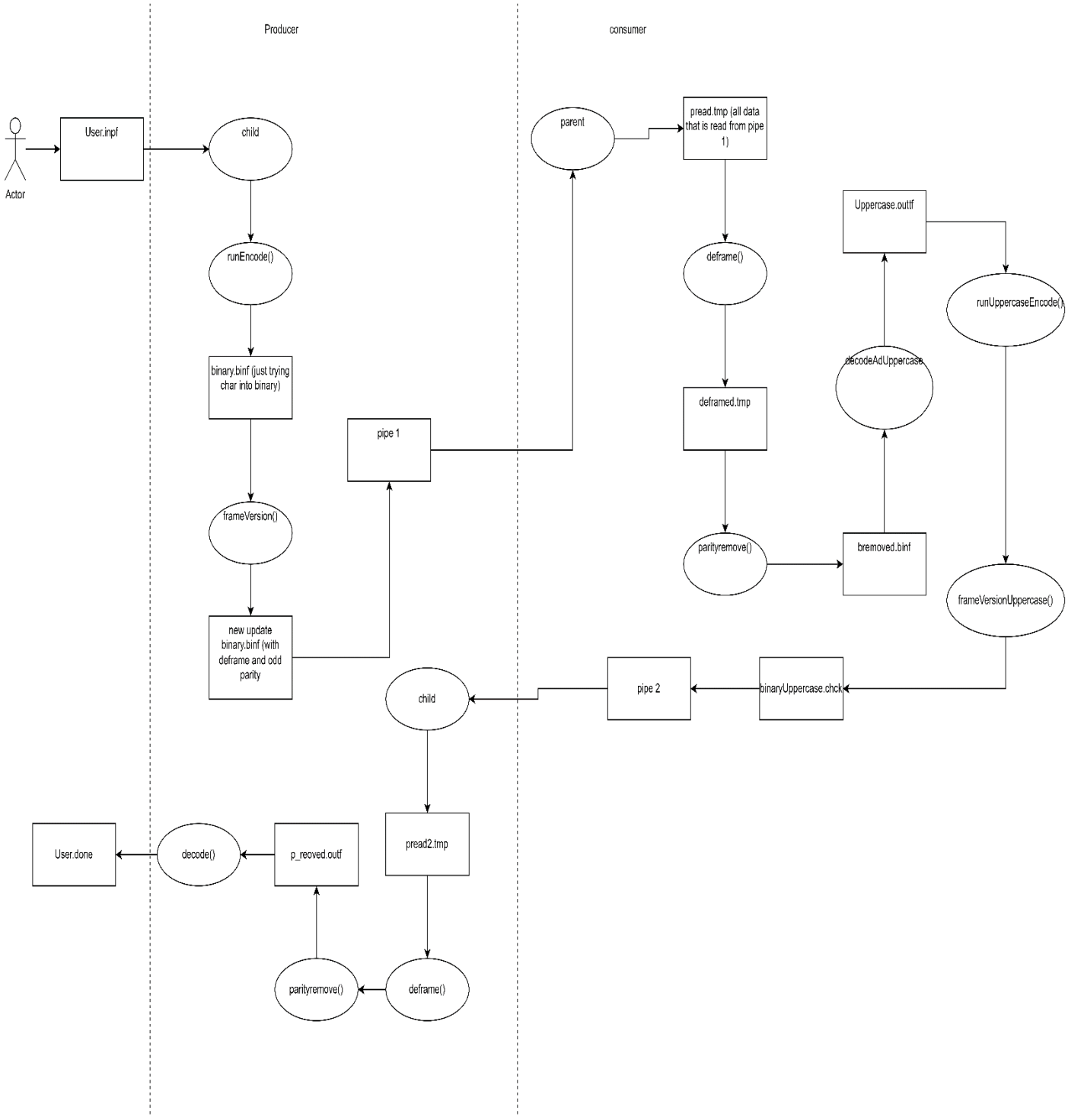
Vepuri, Srujan
UNIVERSITY AT ALBANY

# Contents

# System documentation

About system flow

## A high-level data flow diagram for the system



## A list of routines and their brief descriptions

- Encode – In here we will turn characters to binary and add the parity bit if number in byte is even.
- Decode- In here we will turn binary back characters.
- Priority removed – In here we check and remove binary bit for decode to decode.
- Uppercase – In here we will turn lowercase to uppercase letters.
- Frame – In here we add SYN characters followed by number of characters that will follow bit.
- Deframe – remove SYS characters and decimal number.

## Implementation details

1. We start of from user providing us the “abc.inpf” that our child will take call  encode from our library.
2. From our library we run by execv() encode. Where it will take all the input file that user provide than it will turn every character into binary of 8 bits or 1 byte (ASCII).
3. Then it calls parity bit to add or turn every byte to odd.
4. Then we will call frames and a loop will count till max of 32 characters. Once we have 32 characters or end loop, it will go head write 22 22 in binary and followed by number of characters that we are going place than we will place the characters we counted.
5. Then we read all bits than write to our pipe 1
6. Then from parent we will read all bits and put on temp files.
7. First, we need to deframe so we call execv () that will run start deframe
    a. Inside this c program we first check the 16 bits or 2 bytes to make sure that the binary value is 22.
    b. Once it is than we go read the next 8 bits or 1 byte than we will turn into int number (also check for the number parity bit).
    c. Then we will make any array base on that number an read all the binary and write to deframe file.
8. Once we remove our frames than we will check for our parity bit for each character’s binary. If it is than will remove that or change to parity bit to 0
9. We are now ready to decode. Once decode we will also convert any alphabet to uppercase if not already uppercase.

10. Then we will repeat step from 2, 3 and 4.
11. Now we will write to pipe 2 back our child
12. Then our child will write to temp files
13. We again run our deframes to remove frames and make array size based on the binary number and write to a temp file like from steps 7 to 8
14. Now we will decode and make new file “abc.done” and write all binary that turn into characters.

# Test Documentation

In this we will learn How we test our code

## How I tested my code?

1. My Frist test I stared with very small sentences

    a. “Joke:

    Question: Is Windows a virus?”

2. Then I make sure the “user.done” file has everything. Every character in that done file. I also make sure. I also tested every file that was made by going online checking binary converts. To make sure that it is working correctly.
3. After than then I also ran diff commands with input and done files.
4. I also tested execv, how what path my code takes. I also check how bits are based to my pipes and how bits did my encode had vs how bits my decode() decode.

## My second Test

For my second Test I copied I copied all the input form the pdf project 2 “Is Windows a virus”

Than I ran with lot problems.

user.inpf

Please Look user.inpf for my other test. I used. After fixing those problems. It worked.

## Test set

1. 

    binary.binf

2.  bremoved.binf

3.  p_removed.outf

4.  uppercase.outf

5.  binaryuppercase.chck

6.  user.done

# User documentation

In here it will help user on how or what should do run my code

## How to run your program

1. First you need to compile all my programs that have been used. They need to name the exact program output file need to match what every is the echdec.c libary
    a. gcc deframe.c -o d;
    b. gcc deframeUppercase.c -o du;
    c. gcc framing.c -o f;
    d. gcc framingUppercase.c -o fu;
    e. gcc encode.c -o c;
    f. gcc encodeuppercase.c -o cu;
2. Once you complied those code than go head run my porudcerconsumer.c with any name output program file.

## Describe parameter

User must provide a “user.inpf” file with some characters filled in it. Or name change the name in encode.c file to what user want the files to be read.