Sistemas Operativos

Departamento de Computación – FCEyN – UBA
Primer cuatrimestre de 2019

Nombre y apellido: SANTIAGO FESTINI
Nº orden: 47 L.U.: 317/17 Cant. hojas: 4

Corrigió: Franco

Segundo parcial – 13/6 - Primer cuatrimestre de 2019

1	2	3	4	Nota
B	B	B	B	(A)

ACLARACIONES: 1) **Numere** las hojas entregadas. Esta hoja se entrega y es la hoja cero. Complete en la primera hoja la cantidad total de hojas entregadas (sin contar el enunciado). 2) Realice cada ejercicio en **hojas separadas** y escriba **nombre, apellido y L.U. en cada una**. 3) Cada ejercicio se califica con **Bien**, **Regular** o **Mal**. La división de los ejercicios en incisos es meramente orientativa. Los ejercicios se califican globalmente. El parcial se aprueba con 2 ejercicios bien y a lo sumo 1 mal/incompleto. 4) El parcial **NO** es a libro abierto. 5) **Justifique adecuadamente cada una de sus respuestas.**

Ejercicio 1.

Un usuario borra el historial de un chat confidencial que participó con el comando `rm` sobre un filesystem de tipo Ext2. Sin embargo, un experto en seguridad luego es capaz de recuperar la conversación.

a) Analice cómo puede ocurrir esto y explique en detalle cómo implementar un borrado eficiente en tiempo como el que realiza `rm` y qué estructuras del filesystem debería modificar y por qué.
b) Explique en detalle cómo implementar un borrado seguro que no permita recuperar la conversación y qué estructuras del filesystem debe modificar y por qué.

Ejercicio 2.

Considere los siguientes escenarios de E/S en un computador tipo PC.

a) Un ratón utilizado con una interfaz gráfica de usuario.
b) Una unidad de cinta en un sistema operativo multitarea.
c) Una unidad de disco que contiene archivos de usuario.
d) Una tarjeta gráfica con conexión directa al bus.

Para cada uno de estos escenarios de E/S, ¿cuál diseño convendría para el sistema operativo: buffer en memoria, spooling, o una combinación? ¿Qué sistema de E/S sería utilizaría: polling, manejo de interrupciones o DMA?

Ejercicio 3.

Asuma una versión del compilador que *pushca* las variables locales de las funciones en el stack en orden inverso a donde son declaradas. Es decir, las últimas variables locales son las primeras en ser pusheadas y -por ende- le corresponden las posiciones de memoria más altas del stack.

a) Para el siguiente fragmento de código en C, indique si es vulnerable, y cómo solucionaría esta situación.
b) Si se intercambian las líneas A y B, ¿es o no vulnerable?

```
int mail_auth(char *mecanismo)
{
        char tmp[LONG_CORREO_TMP]; // Linea A
        AUTENTICADOR *auth; // Linea B

        /* convierte a mayusculas el nombre del mecanismo */
        ucase(strcpy(tmp, mecanismo));
        for(auth = autenticadorescorreo; auth, auth = auth->next)
                if (auth->server && strcmp(auth->name, tmp))
                        return 1;
        return NIL;     /* no se encontro el autenticador */
}
```

Ejercicio 4.

En el contexto de los sistemas distribuidos, relacione el mecanismo de consenso de blockchain con el consenso requerido para resolver el problema de consenso bizantino. Justifique su respuesta en cada caso.

a) ¿Es el mismo tipo de consenso el que requieren ambos ámbitos? ¿Por qué? ¿Cuáles son las similitudes y/o diferencias?
b) Si es el mismo tipo, ¿blockchain resuelve el problema bizantino? Si son diferentes, ¿existe alguna modificación a la estrategia planteada en blockchain para que se adapte al problema bizantino?

(1) FESTINI SANTIAGO 31/17

1- A- [redacted] AL EJECUTARSE EL COMANDO RM, LO QUE SUCEDE ES QUE EL INODO DEL DIRECTORIO BORRA SU PUNTERO AL INODO DEL ARCHIVO BORRADO Y ACTUALIZA SU METADATA, AL MISMO TIEMPO QUE EL ARCHIVO BORRADO DECREMENTA EN 1 SU CONTADOR DE REFERENCIAS (UBICADO EN SU INODO), SI ÉSTE VALE 0, ENTONCES ESTE INODO YA NO ES ACCESIBLE POR NADIE Y SE ENCUENTRA DESHABILITADO, NO TENDRÍA QUE BORRAR SU CONTENIDO ES COSTOSO E INNECESARIO EN LA MAYORÍA DE LOS CASOS YA QUE CUANDO SUCEDA LA CREACIÓN DE OTRO INODO QUE TOME SU BLOQUE VA A SOBREESCRIBIR TODA LA INFORMACIÓN, SEA ESTA VACÍO O DATOS DEL INODO SIN BORRAR. ✓ GRACIAS A ESTO, UN EXPERTO EN SEGURIDAD PUEDE RECORRER LOS INODOS INHABILITADOS DEL SISTEMA EN BUSCA DEL INODO DEL ARCHIVO BORRADO. (ASUMO QUE COMO ERA UN ARCHIVO CONFIDENCIAL ENTONCES NO TENÍA MÁS REFERENCIAS QUE LA BORRADA). EN CASO DE QUE EL INODO YA FUESE SOBREESCRITO, QUEDA COMO ÚLTIMO RECURSO RECORRER TODOS LOS BLOQUES DE DATOS Y TRATAR DE RECUPERAR EL ARCHIVO A FUERZA BRUTA. (ESTO TAMPOCO ES SEGURO YA QUE LOS BLOQUES DE DATOS PUEDEN HABER SIDO REUTILIZADOS) [redacted]

• PARA REALIZAR UN BORRADO EFICIENTE EN TIEMPO COMO RM, UNO DEBE ACCEDER AL INODO DEL DIRECTORIO DONDE SE ENCUENTRA EL ARCHIVO, ESTO ES FÁCIL YA QUE TENEMOS QUE TENER EL PATH DEL ARCHIVO QUE DESEAMOS BORRAR Y ENTONCES SIMPLEMENTE DEBEMOS RECORRER CADA INODO DE [redacted] LOS DIRECTORIOS DEL

↳ y el contenido de los directorios también

PATH PARTIENDO DESDE ROOT O CUAL SEA EL BASE HASTA LLEGAR AL DESEADO, DESDE ÉSTE ACCEDEMOS AL INODO DEL ARCHIVO Y DECREMENTAMOS SU CONTADOR DE REFERENCIA EN 1, DE ÉSTE LLEGAR A 0 ENTONCES LO MARCAMOS COMO INHABILITADO/DISPONIBLE Y LO AGREGAMOS A LA LISTA DE INODOS DISPONIBLES (? (NO ESTOY SEGURO DE ESTO ÚLTIMO, PERO CREO QUE TIENE SENTIDO) Y POR ÚLTIMO BORRARÍA EL PUNTERO AL ARCHIVO EN EL INODO DEL DIRECTORIO.

→ es un bitmap, pero la idea es esa

Hay que marcar como contenido libres los bloques de datos

B_ PARA BORRAR UN ARCHIVO DE FORMA SEGURA Y PARA QUE SEA IRRECUPERABLE, SE HACE LO MISMO QUE EXPLIQUÉ ANTES, PERO ADEMÁS SE DEBE LIMPIAR EL BLOQUE DEL INODO Y SUS BLOQUES DE DATOS UBICANDO 0'S EN O ALGÚN NÚMERO EN SU MEMORIA. ✓

* TANTO EN ESTE, COMO EN EL PROCEDIMIENTO ANTERIOR, LOS BLOQUES DE DATOS DEL ARCHIVO BORRADO (SOLO SI SU REFERENCIA ES 0) DEBEN SER MARCADOS COMO DISPONIBLES) ok ✓

OTRA MODIFICACIÓN PODRÍA SER RECORRER TODOS LOS INODOS DE ARCHIVOS BUSCANDO ALGUNO QUE USE MIS MISMOS BLOQUES DE DATOS Y BORRAR A ESE TAMBIÉN PARA UN BORRADO TOTAL.

REFERENCIE → ¡esto no debería suceder!

(2) FESTINI SANTIAGO 3 II/17

2_A_ [illegible]

A_ MOUSE: USARÍA UN BUFFER EN MEMORIA YA QUE NO TIENE SENTIDO USAR SPOOLING PORQUE EL PROCESAMIENTO DE UN MOUSE ES ALGO RÁPIDO Y ACTIVO ✓ POR OTRO LADO, USARÍA UNA COMBINACIÓN DE POLLING E INTERRUPCIONES. INTERRUPCIONES PARA LOS CLICKS Y PARA EL DESLIZAMIENTO INICIAL LUEGO DEL CUAL PASARÍA A POLLING → Ojo, el mouse es mucho más lento que la CPU. MOMENTÁNEAMENTE HASTA NOTAR QUE EL MOUSE SE ENCUENTRA ESTABLE, MOMENTO EN QUE VOLVERÍA A INTERRUPCIONES. ✓

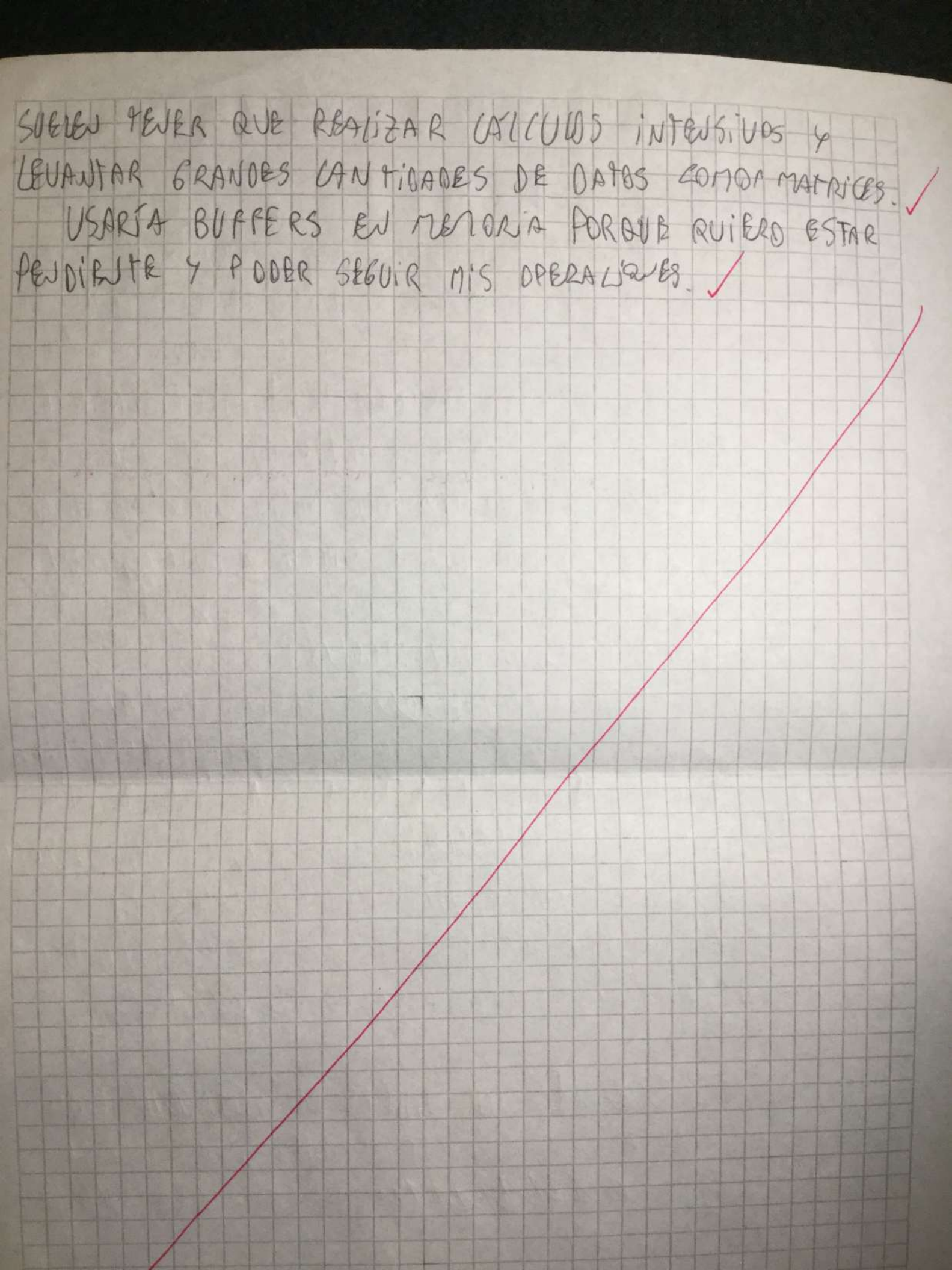
B_ UNIDAD DE CINTA EN S.O. MULTITAREA: DEBIDO A SU LENTITUD Y SU CONDICIÓN MULTITAREA USARÍA SPOOLING CON UN DISPOSITIVO VIRTUAL QUE CONTENGA UN ÚNICO BUFFER PARA TODOS LOS PROCESOS, ADEMÁS, DE SER POSIBLE USARÍA DMA PARA EVITAR EXTRA PROCESAMIENTO DEL CPU, SI NO, USARÍA INTERRUPCIONES PORQUE LAS CINTAS SON UNIDADES DE ALMACENAMIENTO MUY LENTAS POR LO QUE NO ES TERRIBLE RECIBIR INTERRUPCIONES TAN SEPARADAS EN TIEMPO. ✓

C_ UNIDAD DE DISCO: ES UNA UNIDAD DE ALMACENAMIENTO POR LO QUE USARÍA DMA DE SER POSIBLE, SINO, USARÍA INTERRUPCIONES. [illegible] POR OTRO LADO, USARÍA BUFFERS EN MEMORIA. ✓

D_ TARJETA GRÁFICA CON CONEXIÓN DIRECTA AL BUS: USARÍA DMA PORQUE LAS TARJETAS GRÁFICAS

SUELEN TENER QUE REALIZAR CÁLCULOS INTENSIVOS Y LEVANTAR GRANDES CANTIDADES DE DATOS COMO MATRICES. ✓

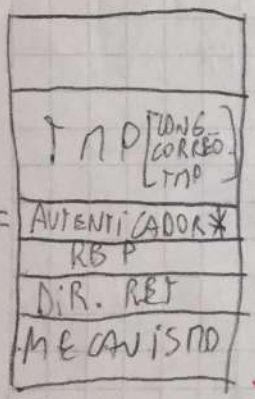
USARÍA BUFFERS EN MEMORIA PORQUE QUIERO ESTAR PENDIENTE Y PODER SEGUIR MIS OPERACIONES. ✓



(3) FESTINI SANTIAGO 31I/17

3- INT MAIL_AUTH (CHAR* MECANISMO) {
CHAR TMP [LONG_CORREO_TMP]; // LÍNEA A
AUTENTICADOR * AUTH; // LÍNEA B

UCASE (STRCOPY (TMP, MECANISMO));
LÍNEA C → FOR (AUTH = AUTENTICADORESCORREO; AUTH, AUTH = AUTH→NEXT) {
IF (AUTH→SERVER && STRCMP(AUTH→NAME, TMP)) {
RETURN 1;
}
}
RETURN NIL;
}



• EL CÓDIGO ANTERIOR ES INSEGURO DEBIDO A LA LLAMADA DE LA FUNCIÓN STRCOPY QUE COPIA EL PARÁMETRO MECANISMO AL BUFFER TEMPORAL SIN RESTRINGIR EL TAMAÑO MÁXIMO DE COPIADO. ✓ ESTO PERMITE GENERAR UN BUFFER OVERFLOW INGRESANDO COMO ENTRADA UNA CADENA DE CARACTERES DE TAMAÑO MAYOR QUE LONG_CORREO_TMP, ASÍ, UN ATACANTE PODRÍA PISAR Y REEMPLAZAR LA DIRECCIÓN DE RETORNO DE LA FUNCIÓN A UNA DE SU CONVENIENCIA. ✓ → ¿cómo?

NOTEMOS QUE AUNQUE EXISTE UN PROBLEMA DE SEGURIDAD, ÉSTE NO PUEDE SER UTILIZADO PARA MODIFICAR A LA VARIABLE AUTH DE UNA MANERA ÚTIL YA QUE AL MOMENTO EN QUE EL ATACANTE PUEDE ALTERAR SU VALOR, AUTH FUE SIMPLEMENTE DECLARADA, NO

INICIALIZADA (ES DECIR QUE ALMACENABA BASURA), POR LO QUE INCLUSO AUNQUE SU VALOR HAYA SIDO REEMPLAZADO, ÉSTE SERÁ REEMPLAZADO OTRA VEZ EN SU INICIALIZACIÓN EN LA LÍNEA C (EL FOR), PISANDO CUALQUIER COSA QUE EL ATACANTE PUEDA HABER PUESTO. DEBIDO A ESTO, LA "SOLUCIÓN" PROPUESTA EN EL INCISO B SIGUE SIN SOLUCIONAR EL PROBLEMA EN CUESTIÓN PORQUE LO ÚNICO QUE ÉSTA CAMBIA ES EL ORDEN EN QUE LAS VARIABLES SON PUSHEADAS A LA PILA IMPIDIENDO ASÍ LA SOBRE-ESCRITURA DE AUTH QUE VIMOS QUE NO ES UN PROBLEMA.

LA SOLUCIÓN MÁS SIMPLE PARA RESOLVER ESTE PROBLEMA ES USAR UNA VARIANTE DE STRCPY QUE SOLO COPIE UNA CIERTA CANTIDAD DE CARACTERES, Y USARLA CON LONG_CORRED_TMP DE CANT. DE CARACTERES MÁXIMOS PARA EVITAR UN BUFFER OVERFLOW.

SI POR ALGÚN MOTIVO ES NECESARIO USAR STRCPY COMO SE ESTA USANDO, O NO SE PUEDE MODIFICAR EL CÓDIGO, O LO QUE FUERA... SE PUEDEN USAR CANARIOS, ÉSTOS SON UNOS VALORES QUE ARRIBA Y ABAJO DE LA DIRECCIÓN DE RETORNO, Y QUE SE ASEGURA DE QUE ESTÉN INTACTOS ANTES DE VOLVER, POR LO QUE RESULTA SER SEGURO ANTE UN BUFFER OVERFLOW PORQUE ESTE NO PUEDE PISARLA SIN PISAR ALGUNO DE ELLOS.

EL COMPILADOR UBICA

LA DIR. DE RETORNO

LOS CANARIOS

(4) FESTINI SANTIAGO 31/1/17

4_ AUNQUE COMPARTEN CIERTAS CARACTERÍSTICAS, EL MECANISMO DE CONSENSO DE LA BLOCKCHAIN ES DISTINTO/ NO APLICA PARA EL PROBLEMA DE CONSENSO BIZANTINO.

SIMILITUDES: - AMBOS SE BASAN EN PONERSE DE ACUERDO EN ALGO, MEDIANTE COMUNICACIONES QUE PUEDEN FALLAR. ✓

- DIFERENCIAS: · EL PROBLEMA BIZANTINO CONSISTE EN ASEGURARSE QUE UN MENSAJE LLEGO A TODOS SUS DESTINATARIOS, MIENTRAS QUE EN EL MECANISMO DE LA BLOCKCHAIN, COMO LAS MÁQUINAS PUEDEN SEGUIR SOLAS, ENVÍAN MENSAJES PERO NO SE PREOCUPAN SI EL RESTO LOS RECIBIÓ O NO. EL MENSAJE BIZANTINO ESTA FIJO Y ES IMPORTANTE QUE LLEGUE A TODOS. LOS MENSAJES DE LA BLOCKCHAIN SON VARIABLES (TODOS DISTINTOS DE HECHO) Y BUSCAN ARMAR UNA MISMA CADENA ENTRE TODOS.
- UN DETALLE QUE ES IMPORTANTE NOTAR ES QUE NO IMPORTA SI UN MENSAJE NO LE LLEGA A UN NODO EN EL CONSENSO DE LA BLOCKCHAIN YA QUE CUANDO A ESTE LE LLEGUE OTRO MENSAJE NUEVO, NOTARÁ QUE ESTA ATRASADO Y PUEDE PEDIR LOS NODOS QUE LE FALTAN PARA PONERSE EN LA VERSIÓN ACTUAL DE LA BLOCKCHAIN. ✓

RESUMIENDO, LA MAYOR DIFERENCIA ES QUE EN LA BLOCKCHAIN NO IMPORTA LA PERDIDA DE MENSAJES,

NI LA NO RECEPCIÓN DE BLOQS, DE HECHO ESTA CONTEMPLADO Y PERMITE PONERSE AL DÍA A LOS NODOS QUE SE ATRACARON O FORKEAR Y CREAR UNA VERSIÓN ALTERNATIVA DE LA BLOCKCHAIN, MIENTRAS QUE EN EL PROBLEMA BIZANTINO ES INDISPENSABLE ASEGURARSE QUE EL MENSAJE LLEGO A TODOS. CONTINUAR ✓

B. PARTIENDO DE LA BASE DE QUE NO EXISTE SOLUCIÓN (100% PRECISA Y HASTA AHORA) AL PROBLEMA BIZANTINO VOY A DECIR QUE NO EXISTE MODIFICACIÓN AL CONSENSO DE LA BLOCKCHAIN. QUE LO RESUELVA.

~~SIN EMBARGO, SI EXISTE QUE FUERON ATACADOS~~

ADEMÁS, POR LO MENCIONADO ANTES, EL ENFOQUE DE AMBOS PROBLEMAS ES BASTANTE DIFERENTE Y NO SE ME OCURRE UNA MODIFICACIÓN ~~QUE SE ACERQUE~~ QUE SE ACERQUE AL PROBLEMA DE UNA MANERA SENCILLA. SIENTO QUE APUNTAN A DOS OBJETIVOS DISTINTOS Y SON DIFÍCILES DE MEZCLAR. ✓