Decryptor 3000[1]

Decryptor 3000 (D3) es un sistema de cálculo distribuido para la búsqueda de claves numéricas por un algoritmo de fuerza bruta.

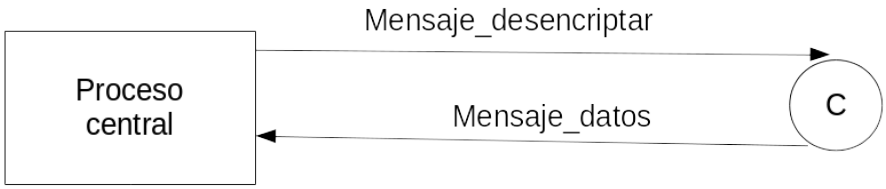
El programa D3 constará de **dos tipos de proceso**:

- Un proceso central gestor de la coordinación (*proceso de E/S, proceso 0*)
- N procesos calculadores auxiliares

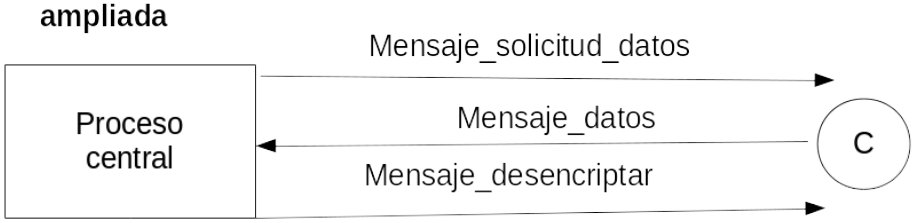
Entre los tipos de proceso tendrán lugar **dos tipos de «transacciones»** (|| proceso de comunicación, diálogo):

- Una *transacción básica*, en la que el proceso central envía los datos necesarios para desencriptar una clave (cf. infra) a un proceso calculador, y en algún momento, este le contesta con los resultados de sus cálculos.
- Una *transacción ampliada*, en la que el proceso central solicita datos de cálculo a un proceso calculador y después inicia una transacción básica.

Transacción básica



Transacción ampliada



En la transacción básica, el proceso central envía los datos necesarios para desencriptar una clave en un mensaje de tipo «Mesaje_desencriptar» (cf. infra) y cuando el proceso ha terminado los cálculos, le responde con un mensaje de tipo «Mensaje_datos» (cf. infra).

Una transacción ampliada tiene lugar cuando

1 Para los primeros 10 compradores se guardará una copia gratuita del proyecto en desarrollo «Encryptor 3000».

a) Un proceso calculador *C* está realizando cálculos para resolver una clave *k*, y uno o más procesos auxiliares A_1, A_2, …, A_n quedan libres y pueden ayudarlo con los cálculos.

b) Una clave calculada por varios procesos ha sido encontrada y es necesario recolectar estadísticas de los otros procesos antes de reasignarles tareas.

En el caso a), el procedimiento a seguir es:

- Enviar al proceso *C* la indicación de que va a haber un cambio en los parámetros de cálculo mediante un mensaje «Mensaje_solicitud_datos».
- El proceso *C* debe entonces enviar los datos de cálculo al proceso central para que la reasignación se haga aprovechando el trabajo ya realizado (las claves ya probadas no se reasignan).
- Comunicar a los procesos los datos para los cálculos mediante una transacción básica.

En el caso, b), cuando uno de los n +1 procesos que se encuentran calculando una clave la encuentra, se lo debe comunicar al proceso central para que este reasigne tareas a todos los procesos que se encontraban calculando esa clave. Esto se realiza mediante una transacción ampliada que primero recolecta estadísticas y luego asigna una nueva clave a los procesos.

Estructuras de datos

Todo esto nos lleva al siguiente punto: el proceso central debe gestionar una tabla de claves y procesos asociados para llevar a cabo una planificación adecuada de todo esto.

Así, una clave debe estar representada por un par (clave, id_clave): es más fácil manejar una tabla mediante identificadores (se puede utilizar el índice de la tabla como identificador). La tabla tiene entonces el aspecto:

Índice = Identificador de clave	Clave	Lista de procesos calculando	Estadísticas
0	000000000	P1, P3	…
1	1111111	P2	…
…	…	…	…
n	kkkkkkkkkk	Pi	…

Por tanto será necesario tener también un identificador para los procesos[2].

Mensajes

El sistema necesitará los siguientes mensajes:

- Mensaje_desencriptar: en él se envían los datos necesarios para desencriptar una clave
  - Clave = (identificador, clave)
  - Rango de cálculo = (valor mínimo, valor máximo)
- Mensaje_datos: contiene los datos resultantes del cálculo de una clave

2 El entorno de MPI proporciona ya un identificador para los procesos; habrá que valorar si es necesario cubrir ese identificador con otro para el programa D3.

- Clave = (identificador, clave)
- Identificador de proceso que ha localizado la clave
- Número de intentos necesarios para encontrarla
- Tiempo empleado para encontrarla

- Mensaje_solicitud_datos: es simplemente una indicación de que se requieren los datos, así que no contiene parámetro alguno.
- Mensaje_finalizar_ejecución: lo manda el proceso central para indicar a un proceso calculador que debe finalizar. Se usa para cerrar el programa D3. No utiliza parámetros.

Estructura de los procesos

Para permitir el funcionamiento tal y como se ha esbozado hasta aquí, un proceso calculador sigue el siguiente esquema:

```
PROCEDURE Calculator {
    DO {
        Clave, etc. ← Recibir_parametros_bloqueante()
        DO  {
            Intentar_desencriptar(Clave)
            IF (Clave_desencriptada) {
                Notificar_clave_encontrada_bloqueante()
                BREAK; // Clave NO disponible
            }

            Mensaje ← Recibir_mensaje_no_bloqueante()
            IF (Hay mensaje) {
                SWITCH (Mensaje) {
                    CASE Mensaje_desencriptar {
                        Clave, etc. ← Get_parametros(Mensaje)
                    }
                    CASE Mensaje_solicitud_datos {
                        Enviar_datos()
                        BREAK; // Clave NO disponible
                    }
                    CASE Mensaje_finalizar_ejecución {
                        BREAK; // NO en ejecución
                    }
```

```
                    }
                }
            } WHILE (Clave disponible)
        } WHILE (En ejecución)
}
```

Por su parte, el proceso central lleva a cabo las siguientes operaciones (tras haber establecido los datos de su entorno – generado claves, etc. –):

```
PROCEDURE Proceso_central {
        …
        WHILE (Claves sin procesos asociados) {
            p ← Seleccionar_proceso_libre()
            Asignar_clave(p)
            IF (No quedan claves sin procesos asociados)
                BREAK; // NO quedan claves libres

            Mensaje ← Recibir_mensaje_no_bloqueante()
            IF (Hay mensaje) { // Solo puede ser de datos y debido a clave encontrada
                Guardar_datos(Mensaje)
                Asignar _clave(Mensaje.proceso)
            }
        }
        WHILE (Queden claves) { // Este bucle WHILE creo que es muy optimizable en el reparto de
                                // tareas
            IF (No hay procesos libres)
                Mensaje ← Recibir_mensaje_bloqueante() // Solo puede ser de datos y
                                                       // debido a clave encontrada
            Datos ← Guardar_datos(Mensaje)
            FOR p IN procesos_calculando_clave_encontrada {
                Enviar_mensaje_solicitud_datos(q)
            }
            FOR n IN numero_de_procesos_calculando_clave {
                Datos ← Recibir_respuesta_bloqueante()
            }
            IF (Quedan claves) {
                FOR p IN procesos_calculando_clave_encontrada
```

```
                    Asignar _clave(p)
                }
                ELSE
                    BREAK; // NO quedan claves
            }
            FOR p IN procesos_calculadores_en_el_sistema
                Enviar_mensaje_finalizar_ejecucion()
            // Cálculos finalizados. Falta mostrar estadísticas, etc.
}

PROCEDURE Asignar_clave(Proceso p) {
            IF (Claves sin procesos asociados) {
                c ← Seleccionar_clave_sin_proceso()
                IF (No quedan claves sin procesos asociados)
                    Set (Flag_no_quedan_claves_sin_proceso)
                Enviar_mensaje_desencriptar(p, c, … )
            } ELSE {
                c ← Seleccionar_clave_con_menor_numero_de_procesos_asociados()
                FOR q IN procesos_calculando_clave_seleccionada {
                    Enviar_mensaje_solicitud_datos(q)
                }
                FOR n IN numero_de_procesos_calculando_clave {
                    Datos ← Recibir_respuesta_bloqueante()
                }
                Realizar_calculos_para_distribuir_trabajo_equitativamente()
                FOR q IN procesos_calculando_clave_seleccionada {
                    Enviar_mensaje_desencriptar(q, c, ...)
                }
                Enviar_mensaje_desencriptar(p, c, ...)
            }
}
```

Falta explicitar qué estructuras de datos tendrán que manejar los procesos calculadores, y quizá alguna del proceso central.