Manual técnico del CH máquina

En este manual veremos aspectos más técnicos del proyecto, tales como qué hacen las funciones, cómo lo hacen, qué variables tenemos y cómo funciona todo por dentro:
En nuestro sistema tenemos las siguientes variables globales que están clasificadas con comentarios:

```
//Dispositivos de salida
const screen = document.querySelector(".output-screen-text");
const printer = document.querySelector(".output-printer-text p");

//Entradas y datos
const file = document.querySelector("#file");
const kernel = document.querySelector("#kernel");
console.log(kernel);
const memory = document.querySelector("#memoria");
const initButton = document.querySelector("#init");
const execButton = document.querySelector("#exec");
const quantum = document.querySelector("#quantum");

//Tamaños límite
const KERNEL_DEFAULT = 59;
const MAX_MEMORY = 5100;
let programasCargados = 0;

//Mapa de memoria y otros
const memoryMap = document.querySelector(".memory-map textarea");
const variables = document.querySelector(".variables textarea");
const tags = document.querySelector(".tags textarea");

//Memoria
const mainMemory = []; //memoria principal
const tagsList = []; //lista de etiquetas de los programas
const variablesList = []; //lista de las variables de los programas
let acumulador = 0;
const procesos = []; //lista de procesos
```

Tenemos una lista de “tokens” que nos referencian a nuestras distintas funciones, se usa en los métodos de planificación para saber qué función ejecutar:

```
const tokens = {
  cargue: cargue,
  almacene: almacene,
  nueva: nueva,
  lea: lea,
```

```
  sume: sume,
  reste: reste,
  multiplique: multiplique,
  divida: divida,
  potencia: potencia,
  modulo: modulo,
  concatene: concatene,
  elimine: elimine,
  extraiga: extraiga,
  Y: Y,
  O: O,
  NO: NO,
  muestre: muestre,
  imprima: imprima,
  vaya: vaya,
  vayasi: vayasi,
  etiqueta: etiqueta,
  retorne: retorne,
  xxxx: xxxx,
};
```

Tenemos nuestra función de anexar programas a la memoria que verifica la sintaxis de cada línea y si no se cumple, no se agrega el programa:

```
file.addEventListener("change", cargarArchivo, false);

function cargarArchivo(e) {
 const archivo = e.target.files[0];
 const lector = new FileReader();
 let lines;
 lector.onload = (e) => {
   let contenido = e.target.result;
   lines = contenido.split(/\r\n|\n/);
   cargarPrograma(lines);
 };

 lector.readAsText(archivo);
 alert(`${archivo.name} cargado con éxito`);
}

function cargarPrograma(lineArray) {
 tagsList[programasCargados] = [];
 variablesList[programasCargados] = [];
 procesos[programasCargados] = [];
```

```
 lineArray.forEach((line) => {
   let instruction = line.trim().split(" ")[0];

   mainMemory.push(line.trim());
   memoryMap.textContent += `${line}\n`;
   procesos[programasCargados].push(line.trim());

   if (Object.keys(tokens).includes(instruction)) {
     if (instruction === "etiqueta") {
       tagsList[programasCargados].push(crearEtiqueta(line));
     }
     if (instruction.toLowerCase() === "nueva") {
       variablesList[programasCargados].push(crearVariable(line));
     }
   } else if (line.startsWith("//") || line === "") {
   } else {
     alert(`Error en la línea ${line}, sintaxis incorrecta`);
     throw new Error("Invalid syntax at " + line);
   }
 });

 programasCargados++;
}
```

Vistos estos detalles, tenemos también la función que nos escoge los métodos de planificación de procesos:

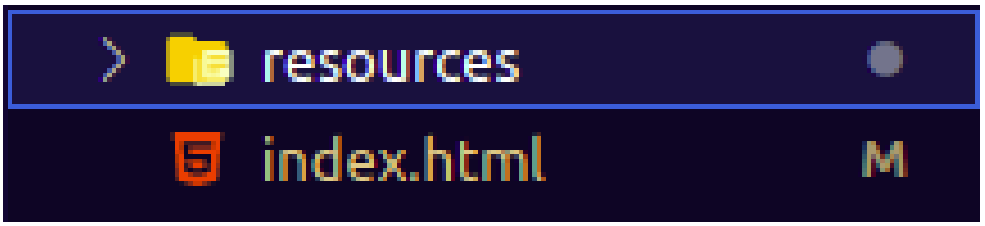
```
function exec(stepByStep) {
 const indiceMetodo = document.querySelector(
   "#metodo-planificacion"
 ).selectedIndex;

 const metodos = {
   0: roundRobin,
   1: SJF,
   2: SJFExpropiativo,
   3: prioridadExpropiativo,
   4: prioridad,
   5: FCFS,
 };

 metodos[indiceMetodo](stepByStep);
}
```

De acá entonces, se ejecutarán los métodos según el usuario escoja.

El proyecto se abre simplemente descomprimiéndolo y abriendo el archivo index.html en su navegador de preferencia (exceptuando versiones viejas de Internet Explorer y Safari):



Este código también se podrá bajar de mi repositorio de GitHub: https://github.com/Ramdhei-codes/ch-maquina como archivo .zip o como repositorio local (Consulte tutoriales de Git y GitHub).