



Instituto Tecnológico de Costa Rica

Escuela de Computación

Principios de Sistemas Operativos GR2

Proyecto #1

Documento de bitácora

Profesora:

Erika Marin Schumann

Integrantes

Moisés Higuerey Hernández (2018092411)

Josue Alvarado Mares (2018143069)

Valeria Quesada Benavides (2018085308)

Fecha de Entrega:

12/4/2021

I Semestre 2022

Primera reunión grupal

Día: 24/03/22

Horas: 7:15pm-8:30pm

Participantes: Josue Alvarado, Moises Higuerey y Valeria Quesada

Objetivo: Inicio del trabajo

Actividad	Hora
Lectura y análisis de las indicaciones en donde se concluye que: <ul style="list-style-type: none">● Hay un cliente automático y otro manual.● El cliente manual inserta un archivo mientras que el automático utiliza randoms entre valores para sus llamadas.● Se deben crear pThreads para cada proceso en el cliente así como Job Scheduler y el CPU Scheduler.● Se debe realizar investigaciones ya que ninguno de los integrantes ha utilizado C en Linux.	7:20pm
Asignación de próximas tareas: <ul style="list-style-type: none">● Para todos los integrantes:<ul style="list-style-type: none">○ Recordar programación de C en Linux○ Investigar cómo realizar pthreads○ Investigar cómo realizar sockets	8:15pm

Asignación de actividades

Día: 26/03/22

Horas: 7:15pm-7:23pm

Participantes: Josue Alvarado, Moises Higuerey y Valeria Quesada

Objetivo: Asignación de pequeñas tareas

Actividad	Hora
Asignación de actividades: <ul style="list-style-type: none">● Josue:<ul style="list-style-type: none">○ Creación y manejo de pThreads.● Moises:<ul style="list-style-type: none">○ Lectura de archivos.● Valeria:<ul style="list-style-type: none">○ Creación y manejo de sockets.	7:17pm

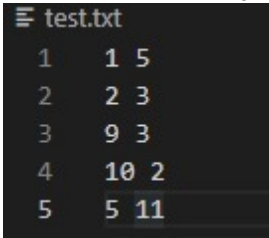
Creación de lectura de archivo

Día: 29/03/22

Horas: 1:00pm-2:55pm

Participantes: Moises Higuerey

Objetivo: Leer los procesos de un txt

Actividad	Hora
Investigación de lectura de archivos en C. <ul style="list-style-type: none">• Leer un archivo en C (1:02pm)<ul style="list-style-type: none">○ Se requiere de fopen para abrir un archivo.○ getline() lee línea de un archivo.○ fclose() se encarga de cerrar el archivo.• Leer archivo línea por línea usando fscanf en C (1:20pm):<ul style="list-style-type: none">○ fscanf permite leer palabra por palabra y línea por línea.	1:02pm
Creación archivo ejemplo. 	1:30pm
Creación de código: <ul style="list-style-type: none">• No hubo errores.	1:35pm

Creación de sockets

Día: 29/03/22

Horas: 2:23pm-4:01pm

Participantes: Valeria Quesada

Objetivo: Conectar cliente con servidor por medio de sockets

Actividad	Hora
Investigación de sockets en C. <ul style="list-style-type: none">• Sockets programming C (2:25pm):<ul style="list-style-type: none">○ Se debe crear un socket en forma de entero.○ La función de bind vincula el socket a la dirección y el número de puerto especificados en la estructura de datos.○ Listen es utilizado para escuchar.○ Accept se encarga de extraer la conexión. Se encuentra en el servidor.○ El cliente crea una conexión con los sockets por medio de connect.• Cliente-Servidor en Linux C (2:45pm):	2:23pm

<ul style="list-style-type: none"> ○ El read se utiliza para leer lo que llega del cliente y servidor. ○ Write se utiliza para enviar un mensaje. ○ Existe multiprogramación. ○ Se necesita un puerto y un IP para establecer conexión. 	
<p>Implementación de código:</p> <ul style="list-style-type: none"> • Se crea un archivo de servidor y otro de cliente. • En el server se hace la creación del socket, el bind de conexión, la opción de aceptar clientes así como la opción de escuchar. <pre> socket_desc = socket(AF_INET, SOCK_STREAM, 0); bind(socket_desc, (struct sockaddr *)&server, sizeof(server)) listen(socket_desc, 3); new_socket = accept(socket_desc, (struct sockaddr *)&client, (socklen_t *)&c) (read_size = recv(new_socket, buffer1, 2000, 0)) </pre> <ul style="list-style-type: none"> • En el caso del cliente se crea el socket, se calcula la dirección y se crea la conexión. <pre> sock = socket(AF_INET, SOCK_STREAM, 0) inet_pton(AF_INET, "127.0.0.1", &serv_addr.sin_addr) connect(sock, (struct sockaddr *)&serv_addr, sizeof(serv_addr)) </pre>	2:57pm

Creación de pthreads

Día: 29/03/22

Horas: 3:08pm-3:56pm

Participantes: Josue Alvarado

Objetivo: Creación de funciones básicas para hilos.

Actividad	Hora
<p>Investigación de pThreads en C de Linux.</p> <ul style="list-style-type: none"> • Creación de pThreads (3:10pm): <ul style="list-style-type: none"> ○ Para crear un pThread se debe inicializar. ○ Para crear un pThread se utiliza la función pthread_create. En donde se utiliza el identificador, los atributos, la función a utilizar, y sus parámetros (son nombrados al final como otro parámetro del pThread). • Utilización de pthread join en C (3:31pm): <ul style="list-style-type: none"> ○ Para esperar que un hilo termine se utiliza pthread_join. 	3:08pm
<p>Implementación de código:</p> <ul style="list-style-type: none"> • Se creó un ejemplo en el cliente. 	3:33pm

<pre>pthread_create(&thrd, NULL, sendProcessSocket, (void *)msg);</pre> <ul style="list-style-type: none"> • sendProcessSocket solo posee una impresión para prueba, sin embargo el hilo nunca se detiene. • Se observan las páginas previamente mencionadas (3:45pm) y se agrega un join. <pre>pthread_join(thrd, NULL);</pre> <ul style="list-style-type: none"> • La prueba pasa. 	
---	--

Revisión de sockets

Día: 29/03/22

Horas: 6:02pm-8:25pm

Participantes: Valeria Quesada

Objetivo: Revisión de sockets

Actividad	Hora
<p>Revisión de sockets:</p> <ul style="list-style-type: none"> • Se corre el server (6:02pm). • El del server bind falla: actualmente se encuentra en uso (6:03pm). Por lo que se investigó <ul style="list-style-type: none"> ◦ Error en el bind (6:05pm): No se encontró coincidencia en los errores. ◦ Error en bind (6:13pm): Se realiza la respectiva prueba. El firewall no lo bloquea. ◦ Manual de Linux - bind (6:27pm): No se encontró razón, se sigue lo requerido. ◦ Bind falla: dirección en uso (6:50pm): Los cambios no funcionaron. ◦ Bind falla: opciones (7:19pm): Se decide no utilizar ninguna de las opciones ya que no parecen ser el problema. ◦ Cómo solucionar el fallo del bind (7:31pm): Se realizó un análisis pero no se hicieron pruebas. ◦ Bind falla con error 98 (8:08pm): Se decide obtener el mensaje de error por medio de strerror(). Obteniendo como respuesta el error 98. El puerto se encontraba malo, se cambió el anterior (80) por (8080). Dando un correcto funcionamiento. • Se corre el cliente (8:21pm). • El bind del cliente falla (8:22pm): actualmente se encuentra en uso. Por lo que se decidió cambiar el puerto. Dando un correcto funcionamiento. • Se corre el server y el cliente (8:23pm), funcionan de manera correcta. 	6:02pm

Implementación de algoritmos.

Día: 30/03/22

Horas: 6:00pm-9:00pm

Participantes: Josue Alvarado, Moises Higuerey y Valeria Quesada

Objetivo: Implementación de algoritmos.

Actividad	Hora
<p>Análisis de algoritmos (6:01pm):</p> <ul style="list-style-type: none">• Análisis de los algoritmos: Todos con excepción del RR son no apropiativos.• Análisis del funcionamiento: La cola se actualiza de manera constante por lo que es muy importante tomar en cuenta la lectura continua de la cola. <p>Creación de los algoritmos (6:33pm):</p> <ul style="list-style-type: none">• Se crea el algoritmo FIFO y se prueba.• Se crea el algoritmo RR y se prueba.• Se crea el algoritmo SJF y se prueba.• Se crea el algoritmo HPF y se prueba.	6:00pm

Asignación de terceras actividades y rendición de cuentas

Día: 03/04/22

Horas: 5:40pm-6:40pm

Participantes: Josue Alvarado, Moises Higuerey y Valeria Quesada

Objetivo: Asignación de pequeñas tareas, rendición de cuentas y creación de listas.

Actividad	Hora
<p>Explicación de actividades previamente asignadas, manera de resolución y problemas encontrados:</p> <ul style="list-style-type: none">• Josue (5:43pm):<ul style="list-style-type: none">◦ Creación y manejo de pThreads. Se realizó su función de creación.• Moises (5:57pm):<ul style="list-style-type: none">◦ Lectura de archivos. Leyó por columnas.• Valeria (6:02pm):<ul style="list-style-type: none">◦ Creación y manejo de sockets. Se creó la conexión de cliente servidor así como envío y recepción de mensajes.	5:40pm
<p>Investigación de listas dinámicas:</p> <ul style="list-style-type: none">• Listas dinámicas en C (6:05pm):<ul style="list-style-type: none">◦ Se requiere hacer una estructura (struct) a la cual acceder. Esta tiene la información de qué tiene cada uno de sus elementos. Además se puede poseer un puntero tipo void al primer y último elemento. Es importante el nodo siguiente para	6:03pm

poder navegar sobre ellas.	
Implementación de código: <ul style="list-style-type: none"> Se creó una estructura para los PCB (6:08pm). Se creó una estructura para los mensajes (6:18pm). Se creó una estructura para el CPU y Job Scheduler (6:22pm). 	6:07pm
Asignación de actividades: <ul style="list-style-type: none"> Josue: <ul style="list-style-type: none"> Creación de pthread por proceso. Moises: <ul style="list-style-type: none"> Función e hilos para el CPU scheduler. Valeria: <ul style="list-style-type: none"> Creación de menús. 	6:37pm

Creación de pthread por proceso

Día: 04/04/22

Horas: 6:02pm-8:13pm

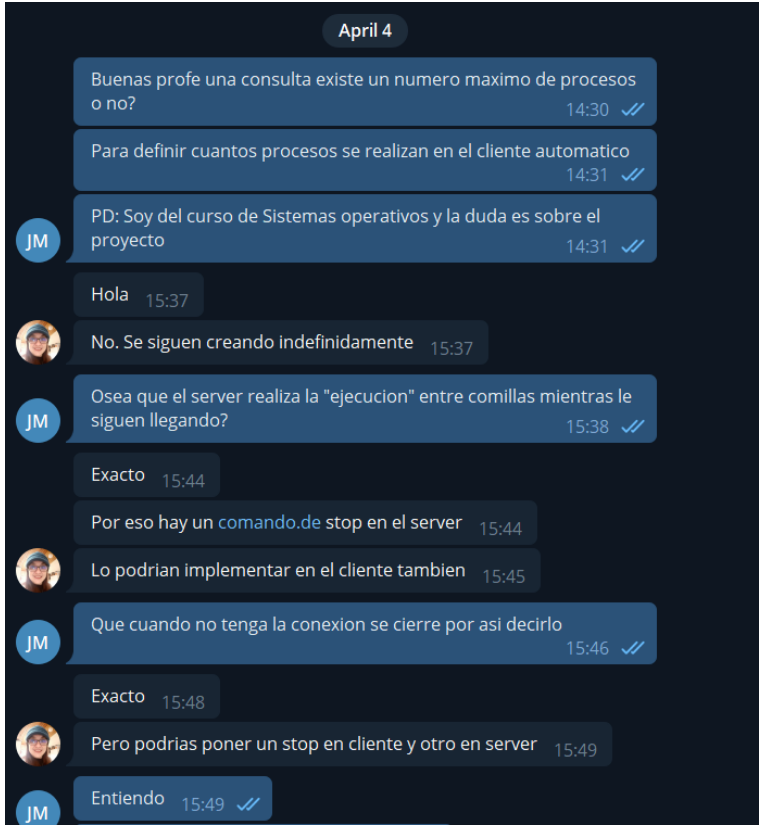
Participantes: Josue Alvarado

Objetivo: Creación de pthread por cada proceso que llega al cliente.

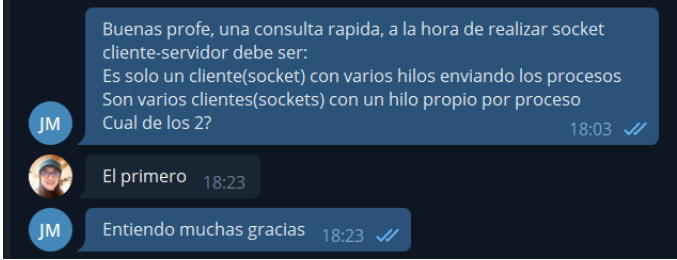
Actividad	Hora
Análisis de la lectura previamente realizada.	6:02pm
Implementación de código en el cliente: <ul style="list-style-type: none"> Creación del hilo por proceso. <pre>pthread_create(&thrd, NULL, sendProcessSocket, (void *)msg);</pre> Se crea la función para el envío de hilos basado en la información del socket que se encuentra en el mismo archivo, por lo que se adapta. De forma en la que se envía y recibe información. <pre>send(my_msg->socket, my_msg->message, strlen(my_msg->message), 0); valread = read(my_msg->socket, buffer, 2000);</pre> Implementación del server: <ul style="list-style-type: none"> El server espera de manera constante los datos del cliente. <pre>// Cycle to continue reading from client socket while ((read_size = recv(new_socket, buffer1, 2000, 0)) > 0) {</pre> Brinda como respuesta al cliente el número del proceso creado con los datos recibidos. <pre>send(new_socket, answer, strlen(answer), 0);</pre> 	6:18pm
Pruebas: <ul style="list-style-type: none"> El cliente envía los datos al servidor y este los recibe para crear el 	

<pre>Procesando Nodo 1: Process ID:1 Burst:1 Priority:5 <-----,> Se ha terminado el proceso #1 con un burst de 1</pre> <p>proceso.</p> <ul style="list-style-type: none"> El servidor envía al cliente el ID del proceso creado. <pre>Se crea el proceso con el PID #1</pre>	
---	--

Consultas realizadas:

Participante	Josue Alvarado
Tema	Cliente automático y comandos del menú
Evidencia	

Participante	Josue Alvarado
Tema	Modelo socket cliente-servidor

Evidencia	
-----------	--

Función e hilos para el CPU scheduler.

Día: 04/04/22

Horas: 6:41pm-9:00pm

Participantes: Moises Higuerey

Objetivo: Función e hilos para el CPU scheduler.

Actividad	Hora
Análisis de la creación de hilos previamente realizada por los integrantes del grupo.	6:41pm
<p>Implementación de código en el server:</p> <ul style="list-style-type: none"> Se crea una estructura con colas indicando si se espera o finaliza un proceso. <pre> struct startCPUScheduler { struct Queue *r; struct Queue *d; int algorithm; int quantum; }; </pre> <ul style="list-style-type: none"> Se crea un hilo de CPU Scheduler <pre> pthread_create(&cpu, NULL, CPU_Scheduler, (void *)cpuData); </pre> <ul style="list-style-type: none"> Se crea una función en donde los procesos ready y terminados se clasifican. Además se toma en cuenta el algoritmo de inicialización dependiendo de lo seleccionado por los usuarios. Lo implementado funciona de manera correcta. 	7:01pm

Creación de menús.

Día: 04/04/22

Horas: 7:00pm-9:00pm

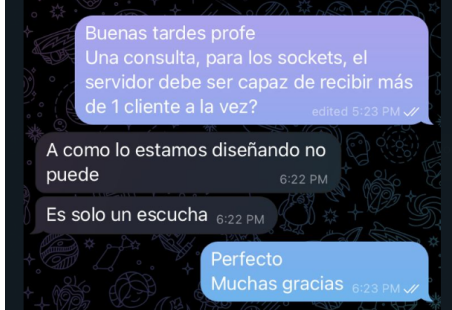
Participantes: Valeria Quesada

Objetivo: Creación de menús cliente servidor.

Actividad	Hora
Análisis de la función e hilos para el CPU scheduler así como el envío de procesos.	6:41pm
<p>Implementación de código en el cliente:</p> <ul style="list-style-type: none">Se crea un menú en el que el cliente selecciona si desea correr el programa con cliente automática (crea solo los procesos) o manual, por medio de números.Dependiendo de la selección anterior, se crean otros dos menús. El primero es manual, en el que se le indica que menciona el nombre del archivo. El segundo es del automático en el que se pide que ponga el tiempo de espera entre la creación de procesos.Al correr el manual, el código falla, por lo que se investiga:<ul style="list-style-type: none">Manual de Linux - scanf (7:43pm): Se utiliza "%s" para tomar cadenas de caracteres. Sin embargo el error continúa.Hacer un scanf() en Linux (7:50pm): Se añade un número entre el "%" y "s" que representa el largo. La prueba funciona: <pre>scanf("%100s", archive);</pre>Se conecta con el main. <p>Implementación del código del servidor:</p> <ul style="list-style-type: none">Se crea un menú para seleccionar el tipo de algoritmo por medio de números indicados.Si el algoritmo es Round Robin se crea una función para que se indique su valor.Se hacen pruebas. No hay errores.Se conecta con el CPU Scheduler y el main.Se hacen pruebas y hay un error en el algoritmo del Round Robin:<ul style="list-style-type: none">Se analiza el código (8:43pm) y se encuentra que no hay validación de nodos por lo que apunta a uno nulo. Se cambia el código:<pre>if (tmp->next != NULL) { // SI HAY SIGUIENTE PREPARA SU PROCESAMIENTO id = tmp->next->process.pId; }</pre>	7:00pm

Consultas realizadas:

Participante	Valeria Qesada
--------------	----------------

Tema	Múltiples conexiones al servidor
Evidencia	 <p>The screenshot shows a WhatsApp chat with a dark background and various icons. The messages are as follows:</p> <ul style="list-style-type: none"> Message 1 (Blue bubble): "Buenas tardes profe. Una consulta, para los sockets, el servidor debe ser capaz de recibir más de 1 cliente a la vez?" (edited 5:23 PM ✓) Message 2 (Grey bubble): "A como lo estamos diseñando no puede" (6:22 PM) Message 3 (Grey bubble): "Es solo un escucha" (6:22 PM) Message 4 (Blue bubble): "Perfecto. Muchas gracias" (6:23 PM ✓)

Creación de menús.

Día: 05/04/22

Horas: 5:40pm-6:08pm

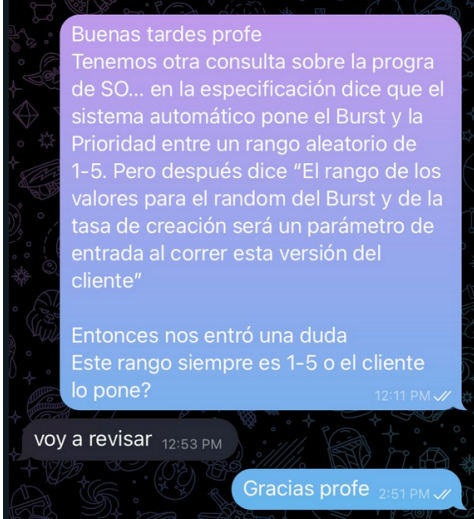
Participantes: Josue Alvarado, Moises Higuerey y Valeria Quesada

Objetivo: Creación de menús cliente servidor.

Actividad	Hora
<p>Explicación de actividades previamente asignadas, manera de resolución y problemas encontrados:</p> <ul style="list-style-type: none"> • Josue (5:43pm): <ul style="list-style-type: none"> ◦ Creación de pThread por cada proceso que llega al cliente. Actualmente no hay errores y se tienen los sleeps solicitados. • Moises (5:57pm): <ul style="list-style-type: none"> ◦ Función e hilos para el CPU scheduler. Actualmente no hay errores y se tienen los sleeps solicitados. • Valeria (6:02pm): <ul style="list-style-type: none"> ◦ Creación de menús. Se creó la conexión con éxito. 	5:40pm
<p>Asignación de actividades:</p> <ul style="list-style-type: none"> • Josue: <ul style="list-style-type: none"> ◦ Creación del Job Scheduler. • Moises: <ul style="list-style-type: none"> ◦ Conexión de datos y pruebas con impresión. • Valeria: <ul style="list-style-type: none"> ◦ Estar escuchando constantemente el teclado. 	6:05pm

Consultas realizadas:

Participante	Valeria Quesada
--------------	-----------------

Tema	Rangos para el modo automático
Evidencia	 <p>Buenas tardes profe Tenemos otra consulta sobre la progra de SO... en la especificación dice que el sistema automático pone el Burst y la Prioridad entre un rango aleatorio de 1-5. Pero después dice "El rango de los valores para el random del Burst y de la tasa de creación será un parámetro de entrada al correr esta versión del cliente"</p> <p>Entonces nos entró una duda Este rango siempre es 1-5 o el cliente lo pone?</p> <p>voy a revisar 12:53 PM</p> <p>Gracias profe 2:51 PM</p>

Creación del Job Scheduler.

Día: 06/04/22

Horas: 5:25pm-7:59pm

Participantes: Josue Alvarado

Objetivo: Se crea el Job Scheduler

Actividad	Hora
Análisis de la función e hilos para el CPU scheduler así como el envío de procesos.	5:26pm
<p>Implementación de código en el server:</p> <ul style="list-style-type: none"> Se crea una estructura para indicar dónde se encuentra el proceso. <pre>struct startJobScheduler { struct Queue *queue; int socket; };</pre> Se crea un hilo de Job Scheduler <pre>pthread_create(&cpu, NULL, CPU_Scheduler, (void *)cpuData);</pre> Se crea una función en donde se lee de forma constante que lleguen procesos, le asigna el pID a los procesos, así como su PCB. <pre>while (new_socket = accept(socket_desc, (struct sockaddr *)&client, (socklen_t *)&c)) while ((read_size = recv(new_socket, buffer1, 2000, 0)) > 0)</pre> 	7:00pm

<pre> struct PCB *process = malloc(sizeof(struct PCB)); process->burst = dataPCB[0]; process->priority = dataPCB[1]; process->pId = pID; process->timeExecute = 0; </pre>	
--	--

Estar escuchando constantemente el teclado

Día: 06/04/22

Horas: 10:00am-11:10pm

Participantes: Valeria Quesada

Objetivo: Escuchar constantemente el IO para evitar interrupciones.

Actividad	Hora
<p>Investigación de cómo escuchar constantemente el teclado:</p> <ul style="list-style-type: none"> • Página de linux - getchar() (10:02am): Se encarga de obtener la tecla seleccionada por el usuario. • Input continuo en C (10:15am): Se prueba la utilización de conio.h. Hay un error debido a que solo se encuentra en Windows, muchos programas de Linux no lo aceptan. • Programa sin bloqueo en Linux (10:29am): Para buscar un input se puede seleccionar select(). Para utilizar ctrl C se debe deshabilitar la función actual y crear una nueva dependiendo de lo que desea • Escuchar al teclado sin detener al programa (10:50am): Se utiliza conio.h con el fin de utilizar kbhit(). • Input del teclado (10:55am): Se utiliza scanf que se encarga de esperar a que el usuario ingrese información. • Detener bucle con tecla C (11:02am): Se utiliza kbhit con conio.h, por lo que no funciona en Linux. 	10:00am

Estar escuchando constantemente el teclado

Día: 06/04/22

Horas: 1:00pm-3:15pm

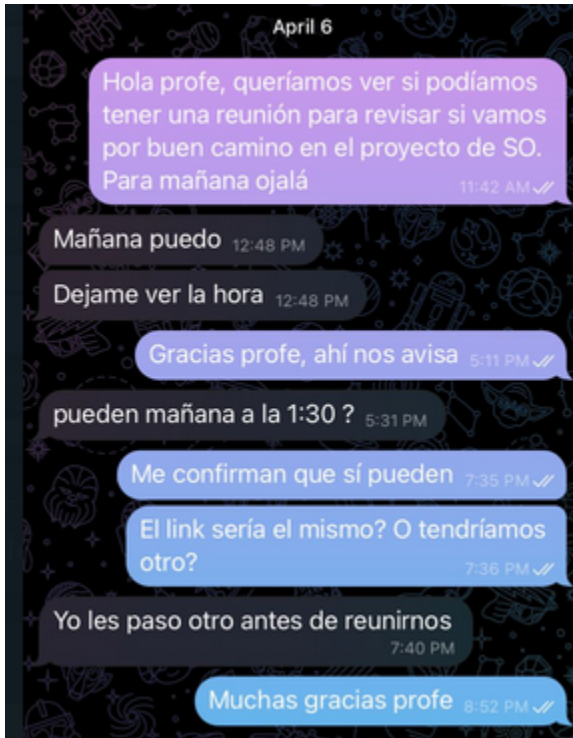
Participantes: Valeria Quesada

Objetivo: Escuchar constantemente el IO para evitar interrupciones.

Actividad	Hora
<p>Investigación de cómo escuchar constantemente el teclado:</p> <ul style="list-style-type: none"> • Estructura terminos (1:00pm): Permite la comunicación de los 	1:00pm

<p>dispositivos de una manera sencilla. Permite leer los caracteres presionados en pantalla, así como las banderas de los estados para la respectiva clasificación. Además permite tener una lectura así como control de la forma en la que se utiliza el <code>getchar()</code>.</p> <ul style="list-style-type: none"> • Salir de ciclos al presionar botón de Linux (1:40pm): Se utiliza la librería <code>termios</code> junto con <code>getchar()</code> para tomar una letra en cualquier momento. Se implementa un código base en un archivo por aparte y funciona. Se sale al presionar la tecla <code>q</code> en minúscula. 	
<p>Implementación del código:</p> <ul style="list-style-type: none"> • Se analiza las funciones en las que el ciclo debe ser implementado. • Se añade la función creada al servidor así como al cliente. Se añade a un ciclo constante de manera en la que se rompe hasta el momento de presionar la tecla <code>q</code>. • Se hacen pruebas y el código funciona. 	2:20pm

Consultas realizadas:

Participante	Valeria Quesada
Tema	Solicitud de reunión.
Evidencia	

Estar escuchando constantemente el teclado.

Día: 07/04/22

Horas: 7:03pm-7:40pm

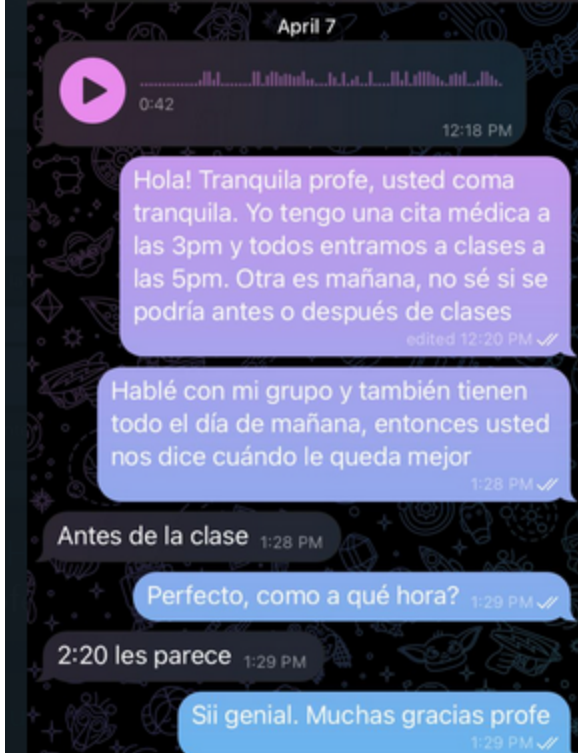
Participantes: Moises Higuerey

Objetivo: Conexión de datos y pruebas con impresión.

Actividad	Hora
<p>Pruebas:</p> <ul style="list-style-type: none">• El servidor se conecta de manera exitosa.• El cliente se conecta con el servidor de manera exitosa y le envía los datos.• Los datos del burst y prioridad son correctamente recibidos por el servidor (Job Scheduler). Se procesa el PID y se guarda en el PCB de cada proceso de manera exitosa. El CPU los procesa de manera exitosa.• Cuando el cliente termina la conexión, el server también lo hace.	7:04pm

Consultas realizadas:

Participante	Valeria Quesada
Tema	Cambio de la fecha de reunión

Evidencia	
-----------	---

Reunión con Prof. Erika

Día: 08/04/22

Horas: 2:20pm - 2:47pm

Participantes estudiantes: Josue Alvarado, Moises Higuerey y Valeria Quesada

Participante profesora: Erika Marín

Objetivo: Aclaración de dudas

Preguntas realizadas por estudiantes	Respuesta de la profesora	Hora
¿El sleep solo detiene el thread? ¿Afecta el resto del programa?	Sí, sí está dentro la función del thread solo debería detenerlo sin afectar el resto del programa.	2:25pm
¿Cómo funciona el burst del programa automático?	El tiempo de espera entre procesos es el que el usuario ingresa.	2:30pm
¿El socket debe ser desde una misma computadora o se pueden conectar varias?	Debería de conectarse con varias. El problema es que actualmente no lo podemos probar.	2:35pm
¿Para matar el programa lo	Sí lo pueden hacer en el server. Yo lo haría	2:38pm

podemos hacer en server?	que el cliente y el server se puedan matar de manera independiente,	
--------------------------	---	--

Algunas recomendaciones	Hora
El servidor sigue escuchando aunque el cliente no exista.	2:43pm
El servidor sigue escuchando aunque no se le mande nada.	2:45pm

Ciclo para escuchar al cliente.

Día: 08/04/22

Horas: 5:00pm-5:11pm

Participantes: Moises Higuerey

Objetivo: Crear un ciclo que siempre escuche al cliente y si se sale que espere a otro. De manera que el socket de servidor no se termine.

Actividad	Hora
Implementación: <ul style="list-style-type: none"> Se crea un while a la hora de crear el socket del servidor. <pre>while (getChar() != 1 & stopServer.stopC != 1)</pre>	5:01pm

Unión de actividades escuchar siempre teclado y aceptar diferentes clientes

Día: 10/04/22

Horas: 1:30pm-2:30pm

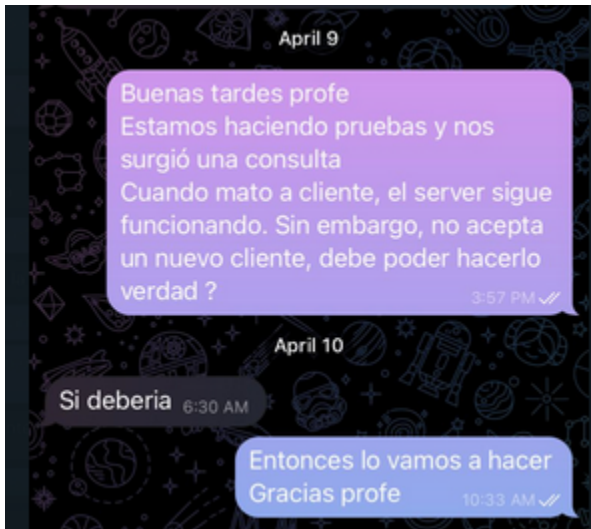
Participantes: Moises Higuerey y Valeria Quesada

Objetivo: Unión de actividades escuchar siempre teclado y aceptar diferentes clientes

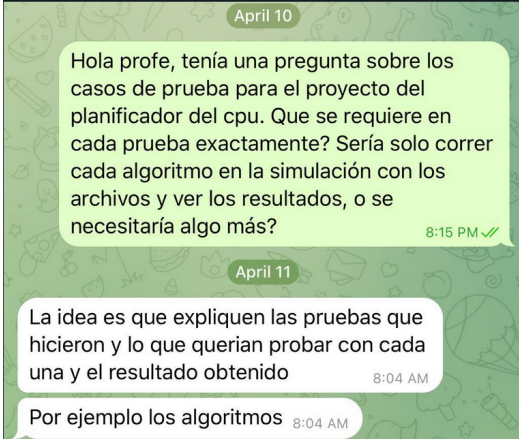
Actividad	Hora
Explicación de actividades previamente asignadas, manera de resolución y problemas encontrados: <ul style="list-style-type: none"> Moises (1:35pm): <ul style="list-style-type: none"> Creación de un loop en el server para escuchar si entra un cliente. No hubo errores. Valeria (1:40pm): <ul style="list-style-type: none"> Creación de función para siempre escuchar el teclado. No hubo errores. 	1:35pm

Unión del siempre escuchar el teclado al loop para aceptar un cliente hasta que el server se caiga: <ul style="list-style-type: none"> Se realizó un push por medio de la plataforma de github para que la rama de Valeria se insertará a la de Moisés y al main. No hubo choques. 	1:45pm
Se hicieron pruebas para verificar que el teclado leyera correctamente: <ul style="list-style-type: none"> Se hicieron prints indicando que se presionó una tecla específica. Hubo un error, por lo que la opción de "getchar()" debió ser sacada de los if y convertida a una variable. Posteriormente se valida la tecla presionada (p para ver la cola del ready y q para parar el socket). 	1:50pm
Se hicieron pruebas para verificar que el cliente pueda salirse y volver a ingresar al servidor: <ul style="list-style-type: none"> No hubo errores, el servidor siempre estuvo escuchando 	2:05pm
Se editó la función encargada de imprimir la cola del ready: <ul style="list-style-type: none"> Se agregaron cambios de línea para una mejor lectura. No hubo errores. 	2:15pm

Consultas realizadas:

Participante	Valeria Quesada
Tema	Varios clientes
Evidencia	 <p>The screenshot shows a WhatsApp chat interface. At the top, it says 'April 9'. A purple message bubble contains the text: 'Buenas tardes profe', 'Estamos haciendo pruebas y nos surgió una consulta', 'Cuando mato a cliente, el server sigue funcionando. Sin embargo, no acepta un nuevo cliente, debe poder hacerlo verdad ?'. Below this, it says 'April 10'. A grey response bubble says 'Si deberia'. A final blue message bubble says 'Entonces lo vamos a hacer' and 'Gracias profe'.</p>

Participante	Moises Higuerey
Tema	Casos de prueba

Evidencia	 <p>April 10</p> <p>Hola profe, tenía una pregunta sobre los casos de prueba para el proyecto del planificador del cpu. Que se requiere en cada prueba exactamente? Sería solo correr cada algoritmo en la simulación con los archivos y ver los resultados, o se necesitaría algo más? 8:15 PM ✓✓</p> <p>April 11</p> <p>La idea es que expliquen las pruebas que hicieron y lo que querian probar con cada una y el resultado obtenido 8:04 AM</p> <p>Por ejemplo los algoritmos 8:04 AM</p>
-----------	---

Tomar tiempo para el cálculo de TAT y WT.

Día: 10/04/22

Horas: 8:00am-9:00am

Participantes: Josue Alvarado

Objetivo: Se calcula el TAT y WT de los procesos.

Actividad	Hora
Investigación: <ul style="list-style-type: none"> • Diferencia entre segundos (8:01am): Se puede utilizar time.h para obtener el tiempo actual. • Diferencia entre tiempos difftime (8:06am): Se utiliza time.h para obtener el tiempo. Lo mejor es utilizar el tiempo local, de manera en la que se toman los segundos actuales de la zona en la que se encuentra. Difftime es una función que calcula la diferencia de tiempos, en este caso, para los segundos. 	8:00am
Implementación del código: <ul style="list-style-type: none"> • Se crean variables que representan el inicio y fin de ejecución de cada proceso. • Se agrega una variable local a la estructura del PCB de manera en la que se sabe el tiempo en el que el proceso llegó y salió. • Se hacen 4 funciones: cálculo del TAT por proceso. cálculo del WT por proceso, promedio de TAT en general y promedio del WT en general. 	8:17am
Búsqueda de errores: <ul style="list-style-type: none"> • Al mostrar el porcentaje de TAT y WT, se encuentra con valores negativos. • Se busca el error y se nota que el tiempo de salida es 0. • Se revisa el código y se ve que no se inicializa, por lo que se le añade a la variable el tiempo actual. • Se hace otra prueba y los tiempos son los correctos. 	8:35

Revisión final.

Día: 11/04/22

Horas: 3:00pm-6:00pm

Participantes: Josue Alvarado, Moises Higuerey y Valeria Quesada

Objetivo: Creación de menús cliente servidor.

Actividad	Hora
<p>Revisión:</p> <ul style="list-style-type: none">• Documentación (3:01pm): Hay comentarios del código en inglés y otros en español, por lo que los de español se cambian al inglés.• Lectura del archivo para encontrar faltantes (3:25pm): Faltaba añadir el CPU ocioso y tabla de resultados al finalizar o salir del programa. Por lo que se añade por medio de impresiones. Al probarlo funciona de manera correcta.• Pruebas (4:00pm):<ul style="list-style-type: none">○ Menús: Se caen al presionar una letra que no es la esperada.○ Impresión de cola ready: Funciona de manera correcta.○ Algoritmos: Funcionan de la manera esperada.○ Impresión final: Funciona de manera correcta○ Interrupción por medio de botones: Funciona de manera correcta.	3:00pm

Consultas realizadas:

Participante	Josue Alvarado
Tema	Manejo de los tiempos (Llegada y salida) por proceso

Evidencia

April 11

JM: Buenas profe no se si pueda hacerle una consulta del pry de SO? 9:49 ✓✓

Si dime 9:59

Vera profe es que estoy haciendo pruebas con fifo sobre el TAT y el WT , pero tengo cierta dudas sobre los datos que estoy teniendo 10:03 ✓✓

Proceso#1
Start time: 0
End time: 2
tat proceso 1: 2 10:03 ✓✓

Proceso #2
Start time: 7
End time: 10
tat proceso 2: 3 10:03 ✓✓

Y en este caso la duda es la siguiente: la diferencia entre el 1 y el 2 puede ser el sleep que tengo en el cliente cuando los envia, pero no se si es lo que quiere 10:05 ✓✓

Porque tengo la opcion de no contar eso y solo contar la duracion que seria
P1
Start: 0
End: 1
Tat: 1
P2
Start: 1
End: 3
Tat: 2 10:06 ✓✓

Josue Gerardo Alvarado Mares
Proceso #2 Start time: 7 End time: 10 tat proceso 2: 3
PD: Estos datos llegan a ser mayores al burst porque cuenta cuando los paso a la cola de terminados, pero no se si quiere lo deje exacto cuando cumple con el burst 10:12 ✓✓

Ya casi te contesto 10:13

Es que ando manejando y no puedo ver el detalle ni sacar los calculos 10:14

JM: Entiendo 😊 10:14 ✓✓

JM: ... 13:30 ✓✓

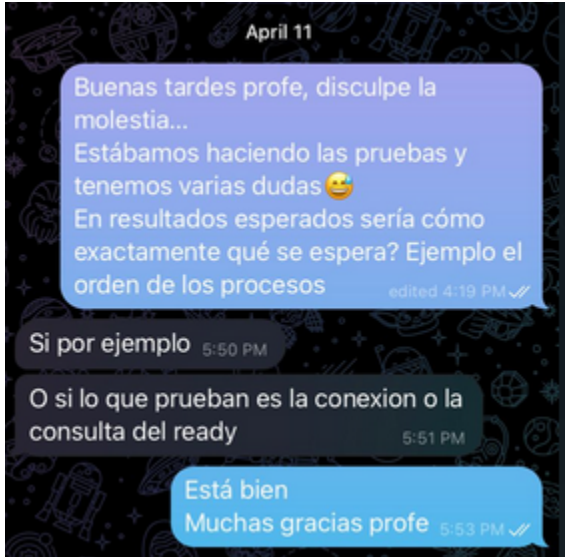
Esta bien como la inicial 17:53

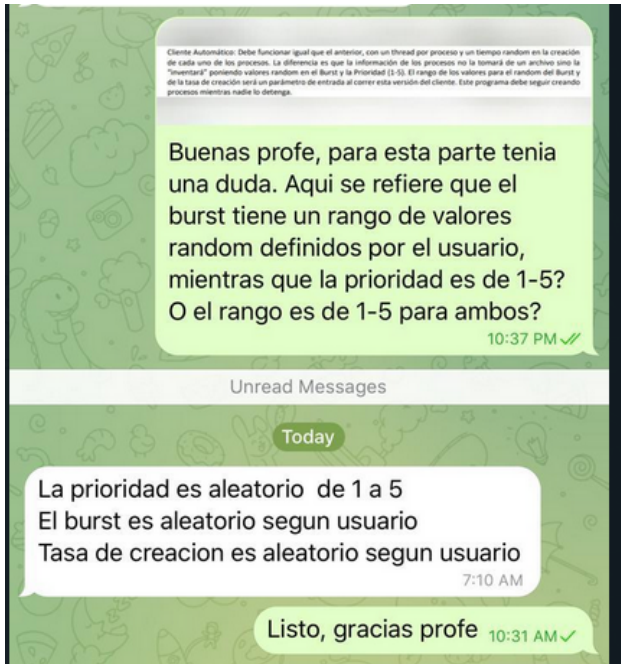
Esta bien el sleep del cliente 17:53

JM: Esta bien muchas gracias 17:54 ✓✓

JM: 😊 Buenas profe queria hacerle una consulta sobre el pry de SO no se, si se puede 18:37 ✓✓

Participante	Valeria Quesada
Tema	Resultados de pruebas

Evidencia	
-----------	---

Participante	Moises Higuerey
Tema	Rangos para el cliente automático.
Evidencia	

Bibliografía

[1] `getchar(3)`: input of char/strings - Linux man page. Linux.die.net. Retrieved 11 April 2022, from <https://linux.die.net/man/3/getchar>.

[2] Khintibidze, L. (2021). *Leer un archivo en C*. Delft Stack. Retrieved 11 April 2022, from <https://www.delftstack.com/es/howto/c/read-file-c/#:~:text=texto%20en%20C.,Usa%20las%20funciones%20fopen%20y%20fread%20para%20leer%20un%20archivo,para%20manipularlo%20como%20sea%20necesario>.

[3] Khintibidze, L. (2021). *Leer archivo línea por línea usando fscanf en C*. Delft Stack. Retrieved 11 April 2022, from <https://www.delftstack.com/es/howto/c/fscanf-line-by-line-in-c/>.

[4] *Socket Programming in C/C++ - GeeksforGeeks*. GeeksforGeeks. (2022). Retrieved 11 April 2022, from <https://www.geeksforgeeks.org/socket-programming-cc/>.

[5] Moon, S. (2020). *How to Code a Server and Client in C with Sockets on Linux - Code Examples - BinaryTides*. BinaryTides. Retrieved 11 April 2022, from <https://www.binarytides.com/server-client-example-c-sockets-linux/>.

[6] Ippolito, G. (2004). *Linux Tutorial: POSIX Threads*. Cs.cmu.edu. Retrieved 11 April 2022, from <https://www.cs.cmu.edu/afs/cs/academic/class/15492-f07/www/pthreads.html>.

[7] Khintibidze, L. (2021). *Utilice la función pthread_join en C*. Delft Stack. Retrieved 11 April 2022, from https://www.delftstack.com/es/howto/c/pthread_join-return-value-in-c/.

[8] use, C., James, B., & Monica, M. (2015). *C Socket program example - Error on binding: Already in use*. Stack Overflow. Retrieved 11 April 2022, from <https://stackoverflow.com/questions/27167434/c-socket-program-example-error-on-binding-already-in-use>.

[9] *Foro de elhacker.NET*. Foro elhacker.NET. (2011). Retrieved 11 April 2022, from https://foro.elhacker.net/programacion_cc/sockets_error_en_bind-t325669.0.html.

[10] Kerrisk, M. (2021). *bind(2) - Linux manual page*. Man7.org. Retrieved 11 April 2022, from <https://man7.org/linux/man-pages/man2/bind.2.html>.

[11] *Bind failed: Dirección ya en uso*. ajaxhispano.com. (2013). Retrieved 11 April 2022, from <https://ajaxhispano.com/ask/bind-failed-direccion-ya-en-uso-88208/>.

[12] *[Solved] Bind failed: Address already in use - Local Coder*. Localcoder.org. Retrieved 11 April 2022, from <https://localcoder.org/bind-failed-address-already-in-use>.

- [13] Davis, H. (2020). *How do I fix bind address already in use?* – QuickAdviser. Quick-adviser.com. Retrieved 11 April 2022, from <https://quick-adviser.com/how-do-i-fix-bind-address-already-in-use/>.
- [14] Van Winkle, L. (2021). *Hands-On Network Programming with C*. Handsonnetworkprogramming.com. Retrieved 11 April 2022, from <https://handsonnetworkprogramming.com/articles/bind-error-98-eaddrinuse-10048-wsaeaddrinuse-address-already-in-use/>.
- [15] *Linked List in C*. DataFlair. (2022). Retrieved 11 April 2022, from <https://data-flair.training/blogs/linked-list-in-c-cpp/>.
- [16] Kerrisk, M. (2021). *scanf(3) - Linux manual page*. Man7.org. Retrieved 11 April 2022, from <https://man7.org/linux/man-pages/man3/scanf.3.html>.
- [17] C, C., Zafar, Z., Barbarosie, A., & Tripathi, R. (2022). *Continuous keyboard input in C*. Stack Overflow. Retrieved 11 April 2022, from <https://stackoverflow.com/questions/19875136/continuous-keyboard-input-in-c>.
- [18] input, C., & Ramsey, N. (2022). *C non-blocking keyboard input*. Stack Overflow. Retrieved 11 April 2022, from <https://stackoverflow.com/questions/448944/c-non-blocking-keyboard-input>.
- [19] C, L. (2016). *Listen for a keypress without stalling the entire program in C*. Stack Overflow. Retrieved 11 April 2022, from <https://stackoverflow.com/questions/35951481/listen-for-a-keypress-without-stalling-the-entire-program-in-c>.
- [20] Brown, B. (1999). *Keyboard Input: scanf()*. Nmu.edu. Retrieved 11 April 2022, from https://www.nmu.edu/Webb/ArchivedHTML/MathComputerScience/c_programming/c_016.htm.
- [21] *struct termios -- data structure containing terminal information*. Mkssoftware.com. Retrieved 11 April 2022, from https://www.mkssoftware.com/docs/man5/struct_termios.5.asp.
- [22] c, b., & Leffler, J. (2013). *breaking loop with keypress in linux c*. Stack Overflow. Retrieved 11 April 2022, from <https://stackoverflow.com/questions/15310848/breaking-loop-with-keypress-in-linux-c>.
- [23] Linux?, H., Starynkevitch, B., & 六四事, C. (2021). *How to print time difference in accuracy of milliseconds and nanoseconds from C in Linux?*. Stack Overflow. Retrieved 11 April 2022, from <https://stackoverflow.com/questions/16275444/how-to-print-time-difference-in-accuracy-of-milliseconds-and-nanoseconds-from-c>.

[24] *difftime* - C++ *Reference*. Cplusplus.com. (2001). Retrieved 11 April 2022, from <http://www.cplusplus.com/reference/ctime/difftime/>.