

Proyecto Primer Parcial - DashPromMQTT v1

Andrea Michelle Vélez Franco

Escuela Superior Politécnica del Litoral

Sistemas Operativos P3

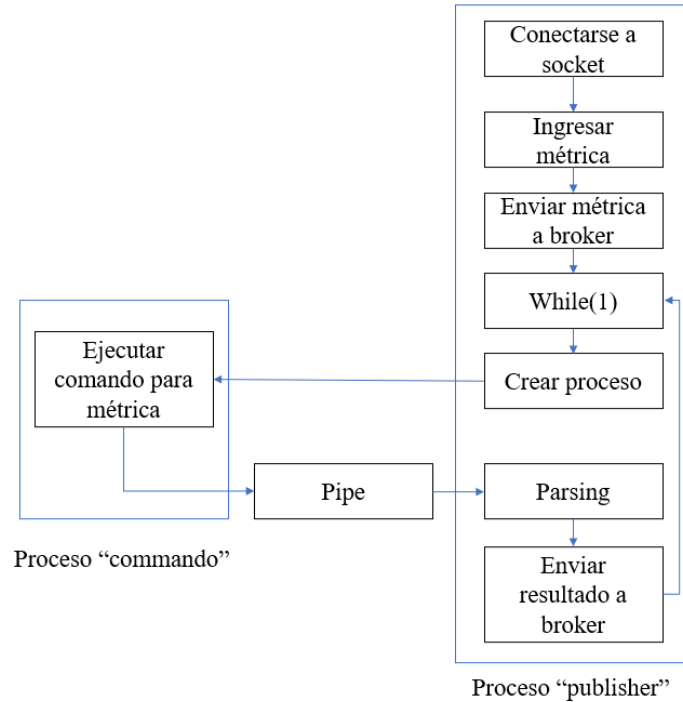
7 de Septiembre de 2022

Índice

1. Diagrama de solución propuesta	3
2. Explicación del código Publisher.c	6
3. Explicación código bróker.c.....	11
4. Explicación código subscriber.c	20
5. Referencias	25

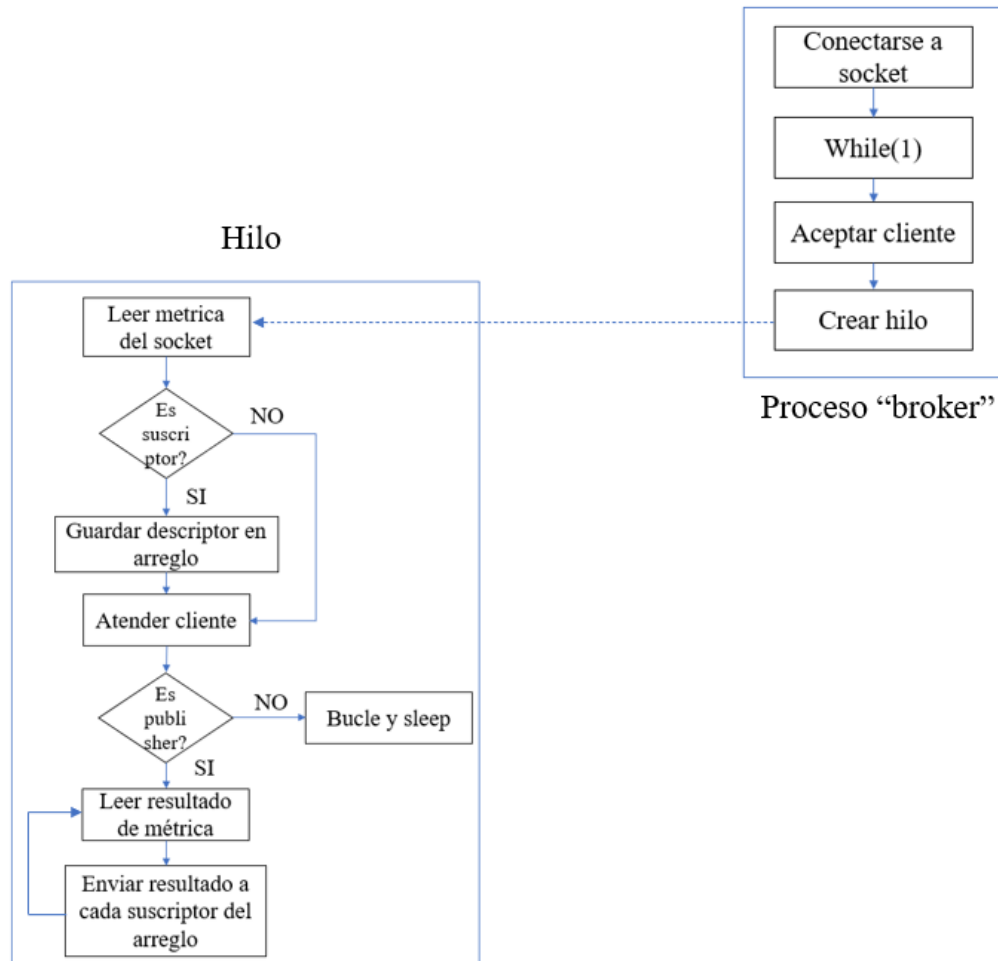
1. Diagrama de solución propuesta

- Diagrama de ejecución del programa Publisher.



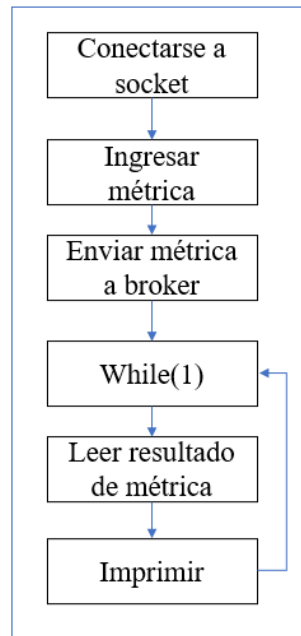
En esta solución, el Publisher ingresa la información de conexión, es decir, el nombre del host. Esta será enviada al bróker para sesgar el envío dependiendo de lo elegido por el suscriptor. El proceso Publisher necesita ejecutar un comando para obtener el valor de la métrica requerida por lo que crea un proceso hijo que mediante `execvp`, ejecuta el comando y redirige la salida al `stdin` del proceso padre. Luego de realizar el parsing de la información, se envía el resultado en modo texto al bróker. Esto es explicado con mayor detalle en el numeral dos de este documento.

- Diagrama de ejecución del programa Broker.



El proceso bróker tendrá un arreglo compartido por métrica para almacenar los descriptors de los suscriptores que solicitan información de esta. Luego, se crean hilos para atender a los clientes. En primer lugar, se lee lo ingresado por el usuario y se identifica si es suscriptor o no. En caso de que lo sea, se almacena su descriptor de conexión con el socket para que después el Publisher pueda iterar sobre ese arreglo y escribir el resultado de la métrica en cada descriptor.

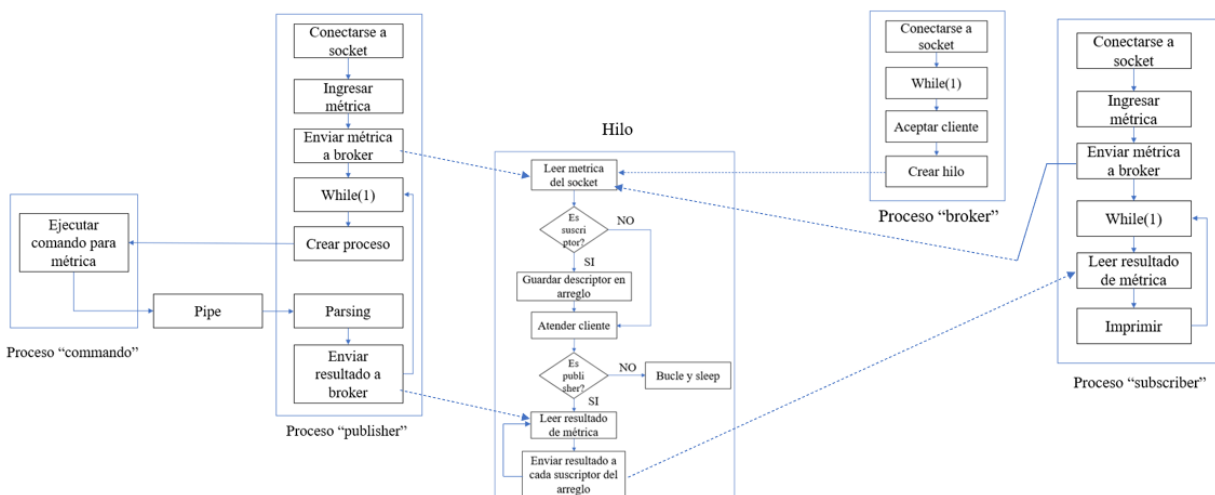
- Diagrama de ejecución del programa Subscriptor



Proceso “subscriber”

En este proceso, el suscriptor únicamente envía la métrica que desea visualizar e ingresa a un bucle para leer lo ingresado por el Broker e imprimirlo en la terminal.

- Diagrama unificado



Este diagrama es una compilación de los mostrados anteriormente incluyendo el envío de los parámetros necesarios mediante las funciones read y write.

2. Explicación del código Publisher.c

La solución planteada se basa en la realización de tres programas que se comunicarán mediante el uso de sockets con un protocolo TCP/IP. Mediante este, se creará un fichero que permita leer y escribir datos enviados con el uso del puerto y la dirección del servidor. Para este modelo, el Publisher realizará la ejecución del comando de Linux mediante procesos, el parsing de la información requerida y enviará el resultado al bróker. Por consiguiente, el bróker enviará la métrica a los suscriptores de ese sistema y ese tópico específico. Así, el bróker será el encargado de redirigir la salida para informar a los clientes con el uso de hilos y un buffer compartido almacenando los descriptores de los suscriptores.

```
int main(int argc, char **argv){
    char *valorfinal;
    char *topico;
    ssize_t n, l, size = 0;
    size_t max = 1024; //MAXLINE;
    int status;
    pid_t pid;

    valorfinal = (char *) calloc(1, 1024);

    //Socket
    int clientfd;
    //Direcciones y puertos
    char *hostname, *port;
    if(argc != 3){
        fprintf(stderr, "uso: %s <hostname> <puerto>\n", argv[0]);
        return -1;
    }else{
        hostname = argv[1];
        port = argv[2];
    }

    //Valida el puerto
    int port_n = atoi(port);
    if(port_n <= 0 || port_n > USHRT_MAX){
        fprintf(stderr, "Puerto: %s invalido. Ingrese un número entre 1 y %d.\n", port, USHRT_MAX);
        return -1;
    }
    //Se conecta al servidor retornando un socket conectado
    clientfd = open_clientfd(hostname, port);

    if(clientfd < 0)
        connection_error(clientfd);

    printf("Conectado exitosamente a %s en el puerto %s.\n", hostname, port);

    print_topicos();
}
```

En esta sección, se realiza todo el proceso de conexión mediante sockets en conjunto con sus validaciones correspondientes en caso de errores. Dentro de los argumentos, al ejecutar el

programa, se debe incluir la dirección y el puerto de la siguiente forma: “./Publisher 127.0.0.1 8090”. Se valida que el puerto sea correcto y luego se conecta al servidor que en este caso será el bróker mediante la función `open_clientfd` que sirve como wrapper de las funciones `socket` y `connect` de modo que envía una petición para conectarse al bróker. Si resulta exitosa la comunicación, se imprimen las opciones de los tópicos a enviar.

```
topico = (char *) calloc(1,1024);
printf("Ingrese el Nombre de host: ");
size = getline(&topico, &max, stdin);
while (1){
    char *valorfinal;
    valorfinal = (char *) calloc(1,1024);
    int fd[2];
    pipe(fd);

    if ((pid = fork()) == 0){
        close(STDOUT);
```

En el código presentado se incluye la función para que el usuario del sistema ingrese su nombre de modo que pueda ser almacenado en un registro y reconocido por los demás usuarios.

Mediante el uso de memoria dinámica, se crea una variable para almacenar el nombre que decida el usuario empezar a enviar haciendo uso del **paso de parámetros desde la terminal**. Es relevante mencionar que esto únicamente será necesario al inicio del programa pues luego ingresará a un bucle para calcular el porcentaje y seguir enviando la información.

Para poder ejecutar los comandos sin afectar al proceso actual, se utiliza un proceso hijo que ejecute el comando por lo que es necesario la **implementación de IPC** con la función `pipe`. De este modo, se redirige el `stdin` y el `stdout` a los descriptores representando al read end y al write end.

```

if ((pid = fork()) == 0){
    close(STDOUT);
    dup(fd[1]); //redirige stdout a la pipe
    close(fd[0]); //hijo no lee del pipe
    close(fd[1]);
    char *comando[] = {"top","-bn2"};

    if (execvp("top",comando) == -1){
        printf("%s: Command not found. \n", comando[0]);
        exit(0);
    }
    exit(0);
} else {
    while ((pid = waitpid(-1, &status,0))>0){
        //
    }
    close(STDIN);
}

```

Como se demuestra en el fragmento de código de la imagen, se crea un proceso hijo mediante la implementación de la función fork para buscar el valor de la métrica simulando la terminal. A continuación, se cierra el descriptor del STDOUT para que no escriba en la terminal y pueda redirigir su resultado al proceso padre con el uso de la función dup que envía el stdout al pipe. También, se cierra el descriptor del read end del pipe creado y el de escritura.

Ahora, debido a que se ha cambiado para recibir varios tópicos independientemente de lo que el usuario desee enviar o no, ya no se incluyen condicionales comparando el tópico ingresado por el usuario con CPU, RAM O SWA de forma que se ejecute el comando necesario. En lo contrario, se ha utilizado la salida del comando top para recopilar toda la información necesaria y enviarla en un string al bróker. Para hallar esta información, se ejecuta el comando: top -bn2 para los valores del consumo de la CPU haciendo dos iteraciones obteniendo mayor precisión. La función execvp facilita la búsqueda al reemplazar el contexto del proceso con la ejecución del comando enviado.


```

} else {
    while ((pid = waitpid(-1, &status, 0)) > 0) {
        //
    }
    close(STDIN);
    dup(fd[0]);
    close(fd[0]);
    close(fd[1]);

    char *linea;
    size_t maximo = 1024;
    linea = (char *) calloc(1, 1024);
    char *segmento;
    int cont = 0;

    //SACA INFO GENERAL DE CPU
    while((getline(&linea, &maximo, stdin)) != -1){
        segmento = strtok(linea, " ");
        if (strcmp(&segmento[0], "%Cpu(s):") == 0){
            cont++;
            if (cont == 2){
                while (segmento != NULL || strcmp(segmento, "(null)") != 0){
                    if (segmento != NULL && strcmp(segmento, "st\n") != 0){
                        strcat(valorfinal, segmento);
                    }
                    if (strcmp(segmento, "st\n") == 0){
                        strcat(valorfinal, "st");
                        break;
                    }
                    segmento = strtok(NULL, " ");
                }
            }
        }
        if (cont == 2){
            break;
        }
    }
}

```

El proceso padre primero espera a que el hijo termine de ejecutar el comando, se cierra el STDIN para poder duplicar el read end del pipe y redireccionarlo como STDIN del padre. Es relevante mencionar que como la salida del comando se encuentra almacenado en el STDIN del padre, se puede utilizar getline para leer cada línea y hacer parsing de lo necesario. En caso de haber ingresado como métrica CPU, se dividen los tokens de la línea hasta llegar a la línea que empiece con cpu del comando top de la segunda iteración y mediante una iteración, se recojan los tokens de la línea. A continuación, se muestra el formato brindado por top para una mejor explicación.

```
top - 01:02:51 up 2:25, 0 users, load average: 0.52, 0.58, 0.59
Tasks: 7 total, 1 running, 6 sleeping, 0 stopped, 0 zombie
%Cpu(s): 6.5 us, 4.3 sy, 0.0 ni, 88.5 id, 0.0 wa, 0.7 hi, 0.0 si, 0.0 st
MiB Mem : 32398.5 total, 22144.6 free, 10029.9 used, 224.0 buff/cache
MiB Swap: 98304.0 total, 98292.9 free, 11.1 used, 22238.0 avail Mem
```

Luego, se almacena toda la línea mediante concat para que puedan existir subniveles, y pueda ser enviada en modo texto como se menciona en las indicaciones del proyecto. Además, se utiliza un contador para obtener el resultado de la segunda iteración del comando top.

```
}
strcat(valorfinal, ";");
//printf("Probando valor final %s\n", valorfinal);
//SACA INFO DE RAM
while((getline(&linea, &maximo, stdin))!=-1){
    segmento = strtok(linea, " ");
    if (strcmp(&segmento[0], "MiB")==0){
        cont++;
        if (cont==3){
            while (segmento!=NULL || strcmp(segmento, "(null)")!=0){
                if (segmento!=NULL && strcmp(segmento, "buff/cache\n")!=0){
                    strcat(valorfinal, segmento);
                }
                if (strcmp(segmento, "buff/cache\n")==0){
                    strcat(valorfinal, "buff/cache");
                    break;
                }
                segmento = strtok(NULL, " ");
            }
        }
    }
    if(cont==3){
        break;
    }
}
//printf("Probando valor final2 %s\n", valorfinal);
strcat(valorfinal, ";");
```

```

//SACA INFO DE SWAP
while((getline(&linea, &maximo, stdin))!=-1){
    segmento = strtok(linea, " ");
    if (strcmp(&segmento[0], "MiB")==0){
        cont++;
        if (cont==4){
            while (segmento!=NULL || strcmp(segmento, "(null)")!=0){
                if (segmento!=NULL){
                    strcat(valorfinal, segmento);
                }
                if (strcmp(segmento, "Mem")==0){
                    //strcat(valorfinal, "Mem");
                    break;
                }
                segmento = strtok(NULL, " ");
            }
        }
        if(cont==4){
            break;
        }
    }
    //printf("prueba final3 %s\n", valorfinal);
    free(linea);
}
printf("%s\n", valorfinal);
n = write(clientfd, valorfinal, 1024); //Envia al servidor
free(valorfinal);
//while(1){
//sleep(2);
//}
}
free(topico);
close(clientfd);
return 0;
}

```

De igual manera, es relevante mencionar que se valida con un contador que sea la línea necesaria debido a que la información de la RAM empieza con MiB y de la SWAP también. Por ende, al haber utilizado dos iteraciones en el comando top y se está utilizando la segunda para mayor precisión, habrá 4 líneas que empiezan con MiB en donde la tercera será la de la RAM y la cuarta la de la SWAP.

Finalmente, en Publisher, se envía la información obtenida en modo texto al bróker mediante el uso de la función write con el descriptor del socket.

3. Explicación código bróker.c

Para el código de Broker, se incluyen arreglos de int como variables globales que almacenarán los descriptors de los sockets con los suscriptores. De este modo, cada métrica tendrá su lista con los

clientes a los que debe enviar la información a las que se suscribieron. De igual manera, se incluyen semáforos para evitar problemas por condición de carrera entre los hilos que serán utilizados para atender a los clientes sin necesidad de un procedimiento secuencial en donde se espere a que se desconecte un cliente para aceptar a otro. La función salir recibe la señal del uso de control c y finaliza el proceso.

```
int bufferHost1[10]= {0};
int bufferCPU[10] = {0}; //se guardan los descriptors de los clientes
int bufferRAM[10] = {0};
int bufferSWA[10] = {0};
int bufferCPUid[10] = {0};
int bufferRAMused[10] = {0};
int bufferSWAused[10] = {0};

sem_t sem;
sem_t sem2;
sem_t sem3;
sem_t sem4;
sem_t sem5;
sem_t sem6;

void atender_cliente(int connfd, char buf[]);

/**
 * Recibe SIGINT, termina ejecución.
 */
void salir(int signal){
    exit(0);
}

void *thread(void* vargp);

int main(int argc, char **argv)
{
    pthread_t tid;
    sem_init(&sem,0,1);
    sem_init(&sem2,0,1);
    sem_init(&sem3,0,1);
    sem_init(&sem4,0,1);
    sem_init(&sem5,0,1);
    sem_init(&sem6,0,1);
```

Se inicializan los semáforos en 1 para ser utilizados como m utex al ingresar a la regi n cr tica y solo uno pueda editar la variable global.

```
if(argv == NULL || argc!=2){
    fprintf(stderr, "uso: %s <puerto>\n", argv[0]);
    return 1;
}
port =argv[1];

//Valida el puerto
int port_n = atoi(port);
if(port_n <= 0 || port_n > USHRT_MAX){
    fprintf(stderr, "Puerto: %s invalido. Ingrese un n mero entre 1 y %d.\n", port, USHRT_MAX);
    return 1;
}
printf("Waiting for connections...\n");

//Registra funcion para se al SIGINT (Ctrl-C)
signal(SIGINT, salir);

//Abre un socket de escucha en port
listenfd = open_listenfd(port);

if(listenfd < 0)
    connection_error(listenfd);

printf("server escuchando en puerto %s...\n", port);
```

Al igual que en el c digo de Publisher, se realizan las conexiones entre el servidor y el cliente con sus validaciones respectivas. Se utiliza la funci n wrapper open_listenfd que realiza las funciones socket, bind y listen en el servidor. Luego, para estar en constante escucha, acepta al cliente y se crea un hilo para que pueda ser atendido envi ndolo a la funci n thread.

```

while (1) {
    clientlen = sizeof(clientaddr);
    connfd = malloc(sizeof(int));
    *connfd = accept(listenfd, (struct sockaddr *)&clientaddr, &clientlen);
    /* Determine the domain name and IP address of the client */
    hp = Gethostbyaddr((const char *)&clientaddr.sin_addr.s_addr,
                      sizeof(clientaddr.sin_addr.s_addr), AF_INET);
    haddrp = inet_ntoa(clientaddr.sin_addr);
    if (hp)
        printf("\n>> Server connected to %s (%s)\n\n", hp->h_name, haddrp);
    else
        printf("server connected to %s\n", haddrp);
    //n = read(connfd, buf, MAXLINE);
    FILE* f = fopen("registro.log", "a");
    if (!f) {
        perror("fopen");
        exit(EXIT_FAILURE);
    }
    char *str1=(char *) calloc(1,1024);
    sprintf(str1, "%s Se ha conectado al broker\n",haddrp);
    fwrite(str1, 1,strlen(str1), f);
    fclose(f);
    pthread_create(&tid, NULL, thread, (void*)connfd);
}

```

Luego de aceptar la conexión con el cliente, se obtiene el nombre de dominio y la IP del cliente de modo que pueda ser almacenada en un registro mediante el uso de fopen y fwrite. Para la implementación de los wildcards y subniveles, se validó la entrada por teclado para ingresar el connfd al arreglo debido. En el caso del wildcard +, se pudo validar fácilmente sin consumir más espacio en memoria al almacenar el connfd en los arreglos de los niveles superiores. Es decir, si se requiere la información de +/free, se almacenará en el arreglo de ambas memorias.

```

void *thread(void* vargp){
    int n;
    int connfd = *((int *)vargp);
    char buf[2000] = {0};
    n = read(connfd, buf, 2000);

    if (strcmp(buf,"host1\n")==0 || strcmp(buf,"host1/#\n")==0){
        int i;
        sem_wait(&sem);
        for(i=0;i<MAXLINE;i++){
            if (!(bufferHost1[i]>0)){
                bufferHost1[i] = connfd;
                break;
            }
        }
        sem_post(&sem);
    }else if(strcmp(buf,"host1/CPU\n")==0 || strcmp(buf,"host1/CPU/#\n")==0){
        int i;
        for(i=0;i<MAXLINE;i++){
            if (!(bufferCPU[i]>0)){
                bufferCPU[i] = connfd;
                break;
            }
        }
    } else if (strcmp(buf,"host1/RAM\n")==0 || strcmp(buf,"host1/RAM/#\n")==0){
        int i;
        sem_wait(&sem2);
        for(i=0;i<MAXLINE;i++){
            if (!(bufferRAM[i]>0)){
                bufferRAM[i] = connfd;
                break;
            }
        }
        sem_post(&sem2);
    }else if (strcmp(buf,"host1/SWA\n")==0 || strcmp(buf,"host1/SWA/#\n")==0){
        int i;
        sem_wait(&sem3);
        for(i=0;i<MAXLINE;i++){
            if (!(bufferSWA[i]>0)){

```

```

        sem_post(&sem3);
    }else if (strcmp(buf,"host1/CPU/idle\n")==0||strcmp(buf,"host1/+/idlePerc\n")==0){
        int i;
        sem_wait(&sem4);
        for(i=0;i<MAXLINE;i++){
            if (!(bufferCPUid[i]>0)){
                bufferCPUid[i] = connfd;
                break;
            }
        }
        sem_post(&sem4);
    }else if (strcmp(buf,"host1/RAM/free\n")==0||strcmp(buf,"host1/+/free\n")==0){
        int i;
        sem_wait(&sem5);
        for(i=0;i<MAXLINE;i++){
            if (!(bufferRAMused[i]>0)){
                bufferRAMused[i] = connfd;
                break;
            }
        }
        sem_post(&sem5);
    }else if (strcmp(buf,"host1/SWA/free\n")==0||strcmp(buf,"host1/+/free\n")==0){
        int i;
        sem_wait(&sem6);
        for(i=0;i<MAXLINE;i++){
            if (!(bufferSWAused[i]>0)){
                bufferSWAused[i] = connfd;
                break;
            }
        }
        sem_post(&sem6);
    }
    pthread_detach(pthread_self());
    free(vargp);
    atender_cliente(connfd,buf);
    close(connfd);
    return NULL;
}

```

Al ingresar en la función thread, el bróker lee del descriptor del socket con el cliente para verificar el ingreso del usuario y definir qué mensajes desea recibir. Por ende, se itera en el buffer definido al inicio del código para agregar al final al nuevo cliente que desea recibir información. De este modo, se almacenan todos los suscriptores permitiendo que cuando ingrese el Publisher, pueda enviar el mensaje de su métrica a cada descriptor del vector. Es importante mencionar que, se utilizan semáforos al agregar un descriptor al arreglo debido a que la posición podría ser sobrescrita con ambos descriptors de los clientes que ingresen y evitar que algún suscriptor sea notificado. Después, se utiliza la función pthread_detach para que luego de su ejecución, los recursos sean

liberados automáticamente y no sea esperado. Así, se llama a la función atender cliente enviando el arreglo con los descriptors de los clientes y el descriptor de la comunicación con el socket para leer y escribir los datos entre los extremos con el bróker como intermediario.

```
void atender_cliente(int connfd, char buf[])
{
    int n;

    if(strcmp(buf,"host1/CPU\n")==0 || strcmp(buf,"host1/RAM\n")==0 || strcmp(buf,"host1/SWA\n")==0 || strcmp(buf,"host1/DB\n")==0)
    {
        //n = read(connfd, buf, MAXLINE);
        FILE* f = fopen("registro.log", "a");
        if (!f) {
            perror("fopen");
            exit(EXIT_FAILURE);
        }
        char *str1=(char *) calloc(1,1024);
        sprintf(str1, "El usuario con connfd %d ha solicitado: %s\n",connfd,buf);
        fwrite(str1, 1,strlen(str1), f);
        fclose(f);
        while(1){
            sleep(1);
        }
    }
    else{
        while(1){
            char *nombre= (char *) calloc(1,100);
            nombre= buf;
            n = read(connfd, buf, 2000);
            if (n>0){
                FILE* f = fopen("registro.log", "a");
                if (!f) {
                    perror("fopen");
                    exit(EXIT_FAILURE);
                }
                char *str1=(char *) calloc(1,1024);
                sprintf(str1, "%s ha enviado %s\n",nombre,buf);
                fwrite(str1, 1,strlen(str1), f);
                fclose(f);
                int i;
                for (i=0;i<10;i++){
                    if(bufferHost1[i]>0){
                        //printf("%s\n",buf);
                        n = write(bufferHost1[i],buf, strlen(buf) + 1);
                    }
                }
            }
        }
    }
}
```

Se verifica que sea el cliente o el publisher, de ser el Publisher, tendrá que leer en su descriptor con el bróker todo el string enviado, iterar entre los suscriptores del arreglo y escribir en aquel descriptor el valor obtenido. Para esto se usan las funciones read y write pues el Publisher lee del descriptor de su conexión con el bróker y escribe en el descriptor del suscriptor de la lista. De igual manera, se registra en el log la información enviada en caso de que sea el Publisher y la métrica que requiere en caso de que sea el cliente. Para evitar problemas el modificar la variable buf que

almacena todo el string, se utiliza strcpy para que no guarde la referencia a memoria de buf y pueda ser utilizada como si fuese el string inicial que envió el Publisher.

```
for (i=0;i<10;i++){
    if(bufferHost1[i]>0){
        //printf("%s\n",buf);
        n = write(bufferHost1[i],buf, strlen(buf) + 1);
    }
    if(bufferCPU[i]>0){
        char *token= strtok(buf,";");
        n = write(bufferCPU[i],token, strlen(token) + 1);
    }
    if(bufferRAM[i]>0){
        char *token= strtok(buf,";");
        token= strtok(NULL,";");
        //printf("%s\n", token);
        n = write(bufferRAM[i],token, strlen(token) + 1);
    }
    if(bufferSWA[i]>0){
        char *token= strtok(buf,";");
        token= strtok(NULL,";");
        token=strtok(NULL,";");
        n = write(bufferSWA[i],token, strlen(token) + 1);
    }
    char stringfree[1024] = "";
    strcpy(stringfree,buf);
    if(bufferRAMused[i]>0){
        if(i==0){
            char *ram= strtok(buf,";");
            //printf("ram1 %s\n",ram);
            ram= strtok(NULL,";");
            //printf("ram2 %s\n",ram);
            ram = strtok(ram, ",");
            ram=strtok(NULL,"");
            //ram=strtok(NULL,"");
            //printf("ram %s\n",ram);
            char *string=(char *) calloc(1,1024);
            sprintf(string, "RAM free: %s\n",ram);
            //printf("string %s\n",string);
            int i;
            for(i=0;i<4;i++){
                if(bufferRAMused[i]>0){
                    n = write(bufferRAMused[i],string, strlen(string) + 1);
                }
            }
        }
    }
}
```

```

        char *string=(char *) calloc(1,1024);
        sprintf(string, "RAM free: %s\n",ram);
        //printf("string %s\n",string);
        int i;
        for(i=0;i<4;i++){
            if(bufferRAMused[i]>0){
                n = write(bufferRAMused[i],string, strlen(string) + 1);
            }
        }
    }
    if(bufferSWAused[i]>0){
        if(i==0){
            if(strcmp(stringfree, "")!=0){
                strcpy(buf,stringfree);
            }
            char *token= strtok(buf,";");
            token= strtok(NULL,";");
            token=strtok(NULL,";");
            token= strtok(token,";");
            token = strtok(NULL, ";");
            //token=strtok(NULL,";");
            //swa=strtok(NULL,";");
            char *string=(char *) calloc(1,1024);
            sprintf(string, "SWAP free: %s\n",token);
            int i;
            for(i=0;i<4;i++){
                if(bufferSWAused[i]>0){
                    n = write(bufferSWAused[i],string, strlen(string) + 1);
                }
            }
        }
    }
    if(bufferCPUid[i]>0){
        if(i==0){
            char *cp= strtok(buf,";");
            //printf("probando buf %s\n en iter %d", buf, i);

            cp= strtok(cp, ",");
            //printf("probando buf %s\n", buf);

```

```

    }
    if(bufferCPUid[i]>0){
        if(i==0){
            char *cp= strtok(buf,"");
            //printf("probando buf %s\n en iter %d", buf, i);

            cp= strtok(cp, ",");
            //printf("probando buf %s\n", buf);
            cp=strtok(NULL,"");
            cp=strtok(NULL,"");
            cp=strtok(NULL,"");
            int i;
            for(i=0;i<4;i++){
                if (bufferCPUid[i]>0){
                    n = write(bufferCPUid[i],cp, strlen(cp) + 1);
                }
            }
            //n = write(bufferCPUid[i],cp, strlen(cp) + 1);
        }
    }
}
memset(buf, 0, MAXLINE); //Encera el buffer

```

Se realiza el proceso mencionado anteriormente para cada una de las métricas a enviar. Por otro lado, en caso de que se esté atendiendo al suscriptor, se incluye un bucle y la función sleep para que esté en constante escucha recibiendo la información y no se cierre el descriptor del socket. En caso de no incluirlo, la función pasaría a close(connfd) y el cliente ya no pueda leer lo escrito por los publishers.

4. Explicación código subscriber.c

En primera instancia, se realiza la conexión con el bróker mediante la dirección y el puerto con la función wrapper open_clientfd mencionada anteriormente de igual manera que en Publisher.

```

int main(int argc, char **argv)
{
    //Socket
    int clientfd;
    //Direcciones y puertos
    char *hostname, *port;

    //Lectura desde consola
    char *linea_consola;
    char read_buffer[MAXLINE + 1] = {0};
    size_t max = MAXLINE;
    ssize_t n, l = 0;

    if(argc != 3){
        fprintf(stderr, "uso: %s <hostname> <puerto>\n", argv[0]);
        return -1;
    }else{
        hostname = argv[1];
        port = argv[2];
    }

    //Valida el puerto
    int port_n = atoi(port);
    if(port_n <= 0 || port_n > USHRT_MAX){
        fprintf(stderr, "Puerto: %s invalido. Ingrese un número entre 1 y %d.\n", port, USHRT_MAX);
        return -1;
    }

    //Se conecta al servidor retornando un socket conectado
    clientfd = open_clientfd(hostname, port);

    if(clientfd < 0)
        connection_error(clientfd);

    printf("Conectado exitosamente a %s en el puerto %s.\n", hostname, port);
}

```

```

#include <pthread.h>
#include <stdio.h>
#include <limits.h>
#include "common.h"

void print_topicos(void);
void print_topicos(void){
    printf("Escoja la metrica que desea recibir, cada subnivel incluirlo con un /\n");
    printf("Metricas:\n");
    printf("(Host#)\t\tRecibir toda la informacion del sistema\n");
    printf("(CPU)\t\tInformacion de la CPU\n");
    printf("(RAM)\t\tInformacion de la memoria RAM\n");
    printf("(SWA)\t\tInformacion de la memoria SWAP\n");
    printf("(+/+)\t\tWildcard de un nivel\n");
    printf("(/#)\t\tWildcard de varios niveles\n");
    printf("\n");
}

int main(int argc, char **argv)
{
    //Socket
    int clientfd;
    //Direcciones y puertos
}

```

En el código presentado se incluye la función para que el usuario del sistema elija qué métrica desea recibir de los publishers. Se incluyeron tres métricas del estado del sistema como: Porcentaje de uso del procesador, porcentaje de memoria libre y usada de la RAM y en la memoria virtual

SWAP. De igual manera, se podrá verificar el porcentaje de idle time del procesador, el espacio libre de la RAM y el espacio libre de la SWAP.

Luego, se le pide al usuario que ingrese el host y el tópico del que desea recibir mensajes y se envía lo elegido mediante la función write al descriptor del socket. Consecuentemente, para poder mostrar lo recibido por el Publisher, se incluye un bucle para que lea del descriptor las métricas enviadas y las muestre con un printf.

```
linea_consola = (char *) calloc(1, MAXLINE);
printf("ingrese el host y la metrica divididos por / >>");
l = getline(&linea_consola, &max, stdin); //lee desde consola, linea_consola es el host y la metrica
n = write(clientfd, linea_consola, l); //Envia al servidor

while(1){
    n=read(clientfd, read_buffer,MAXLINE);
    if (n>0){
        printf("%s\n", read_buffer);
    }
    memset(read_buffer,0,MAXLINE + 1); //Encerar el buffer
}

printf("Desconectando...\n");
free(linea_consola);
close(clientfd);
}
```

Evidencia de ejecución con las siguientes entradas:

- Host1 o Host1/#

```
anvefran@DESKTOP-B4IKBDH: ~/sistemasOp/proyecto2/proyecto
Microsoft Windows [Version 10.0.19844.1889]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Andi_xubuntu2204>
anvefran@DESKTOP-B4IKBDH:~$ cd sistemasOp
anvefran@DESKTOP-B4IKBDH:~/sistemasOp$ cd proyecto2
anvefran@DESKTOP-B4IKBDH:~/sistemasOp/proyecto2$ cd proyecto/
anvefran@DESKTOP-B4IKBDH:~/sistemasOp/proyecto2/proyecto$ ./broker 8090
Waiting for connections...
server escuchando en puerto 8090...

>> Server connected to localhost (127.0.0.1)

>> Server connected to localhost (127.0.0.1)

>> Server connected to localhost (127.0.0.1)

-

anvefran@DESKTOP-B4IKBDH: ~/sistemasOp/proyecto2/proyecto
anvefran@DESKTOP-B4IKBDH:~/sistemasOp/proyecto2/proyecto$ ./publisher 127.0.0.1 8090
Conectado exitosamente a 127.0.0.1 en el puerto 8090.
Ingrese el Nombre de host: host1
%Cpu(s):0.9us,2.2sy,0.0ni,96.7id,0.0wa,0.3hi,0.0si,0.0st;MiBMem:32398.5total,26685.8free,5488.8used,224.0buff/cache;MiBSwap:98304.0total,98291.4free,12.6used.26779.1a
vailMem
%Cpu(s):0.2us,0.8sy,0.0ni,98.8id,0.0wa,0.1hi,0.0si,0.0st;MiBMem:32398.5total,26643.7free,5530.8used,224.0buff/cache;MiBSwap:98304.0total,98291.4free,12.6used.26737.1a
vailMem
%Cpu(s):0.5us,1.0sy,0.0ni,98.3id,0.0wa,0.2hi,0.0si,0.0st;MiBMem:32398.5total,26641.8free,5532.7used,224.0buff/cache;MiBSwap:98304.0total,98291.4free,12.6used.26735.2a
vailMem
%Cpu(s):0.0us,0.1sy,0.0ni,99.8id,0.0wa,0.0hi,0.0si,0.0st;MiBMem:32398.5total,26676.4free,5498.1used,224.0buff/cache;MiBSwap:98304.0total,98291.4free,12.6used.26769.8a
vailMem
%Cpu(s):4.1us,6.5sy,0.0ni,89.0id,0.0wa,0.4hi,0.0si,0.0st;MiBMem:32398.5total,26620.9free,5553.6used,224.0buff/cache;MiBSwap:98304.0total,98291.4free,12.6used.26714.3a
vailMem
-

anvefran@DESKTOP-B4IKBDH: ~/sistemasOp/proyecto2/proyecto
anvefran@DESKTOP-B4IKBDH:~/sistemasOp/proyecto2/proyecto$ ./subscriber 127.0.0.1 8090
Conectado exitosamente a 127.0.0.1 en el puerto 8090.
Escriba la metrica que desea recibir, cada subnivel incluirlo con un /
Metricas:
(Host#) Recibir toda la informacion del sistema
(CPU) Informacion de la CPU
(RAM) Informacion de la memoria RAM
(SWA) Informacion de la memoria SWAP
(/+/-) Wildcard de un nivel
(/#) Wildcard de varios niveles

ingrese el host y la metrica divididos por / >>host1/
%Cpu(s):0.0us,0.1sy,0.0ni,99.8id,0.0wa,0.0hi,0.0si,0.0st;MiBMem:32398.5total,26676.4free,5498.1used,224.0buff/cache;MiBSwap:98304.0total,98291.4free,12.6used.26769.8a
vailMem
%Cpu(s):4.1us,6.5sy,0.0ni,89.0id,0.0wa,0.4hi,0.0si,0.0st;MiBMem:32398.5total,26620.9free,5553.6used,224.0buff/cache;MiBSwap:98304.0total,98291.4free,12.6used.26714.3a
vailMem
```

Al escribir esta entrada, se le envia toda la información del host1, es decir, CPU, RAM y SWAP. El subscriber de la parte inferior derecha llegó después que el subscriber superior por lo que la información de las métricas anteriores no fue recibida.

- Host1/CPU o Host1/CPU/#

```
anvefran@DESKTOP-B4IKBDH: ~/sistemasOp/proyecto2/proyecto
>> Server connected to localhost (127.0.0.1)

>> Server connected to localhost (127.0.0.1)

^Canvefran@DESKTOP-B4IKBDH:~/sistemasOp/proyecto2/proyecto$ ./broker 8090
Waiting for connections...
server escuchando en puerto 8090...

>> Server connected to localhost (127.0.0.1)

>> Server connected to localhost (127.0.0.1)

>> Server connected to localhost (127.0.0.1)

-

anvefran@DESKTOP-B4IKBDH: ~/sistemasOp/proyecto2/proyecto
3free,5586.2used,224.0buff/cache;MiBSwap:98304.0total,98291.4free,12.6used.26681.7a
vailMem
%Cpu(s):3.8us,10.9sy,0.0ni,84.9id,0.0wa,0.5hi,0.0si,0.0st;MiBMem:32398.5total,26633.1free,5541.4used,224.0buff/cache;MiBSwap:98304.0total,98291.4free,12.6used.26726.5
vailMem
%Cpu(s):2.4us,4.8sy,0.0ni,92.4id,0.0wa,0.5hi,0.0si,0.0st;MiBMem:32398.5total,26637.2free,5537.3used,224.0buff/cache;MiBSwap:98304.0total,98291.4free,12.6used.26730.6a
vailMem
^
anvefran@DESKTOP-B4IKBDH:~/sistemasOp/proyecto2/proyecto$ ./subscriber 127.0.0.1 8090
Conectado exitosamente a 127.0.0.1 en el puerto 8090.
Escriba la metrica que desea recibir, cada subnivel incluirlo con un /
Metricas:
(Host#) Recibir toda la informacion del sistema
(CPU) Informacion de la CPU
(RAM) Informacion de la memoria RAM
(SWA) Informacion de la memoria SWAP
(/+/-) Wildcard de un nivel
(/#) Wildcard de varios niveles

ingrese el host y la metrica divididos por / >>host1/CPU
%Cpu(s):0.1us,1.8sy,0.0ni,97.3id,0.0wa,0.7hi,0.0si,0.0st

anvefran@DESKTOP-B4IKBDH: ~/sistemasOp/proyecto2/proyecto
3free,5586.2used,224.0buff/cache;MiBSwap:98304.0total,98291.4free,12.6used.26681.7a
vailMem
%Cpu(s):3.8us,10.9sy,0.0ni,84.9id,0.0wa,0.5hi,0.0si,0.0st;MiBMem:32398.5total,26633.1free,5541.4used,224.0buff/cache;MiBSwap:98304.0total,98291.4free,12.6used.26726.5
vailMem
%Cpu(s):2.4us,4.8sy,0.0ni,92.4id,0.0wa,0.5hi,0.0si,0.0st;MiBMem:32398.5total,26637.2free,5537.3used,224.0buff/cache;MiBSwap:98304.0total,98291.4free,12.6used.26730.6a
vailMem
^
anvefran@DESKTOP-B4IKBDH:~/sistemasOp/proyecto2/proyecto$ ./publisher 127.0.0.1 8090
Conectado exitosamente a 127.0.0.1 en el puerto 8090.
Ingrese el Nombre de host: Host1
%Cpu(s):0.6us,3.2sy,0.0ni,95.6id,0.0wa,0.6hi,0.0si,0.0st;MiBMem:32398.5total,26658.0free,5516.5used,224.0buff/cache;MiBSwap:98304.0total,98291.4free,12.6used.26751.4a
vailMem
%Cpu(s):0.1us,1.8sy,0.0ni,97.3id,0.0wa,0.7hi,0.0si,0.0st;MiBMem:32398.5total,26659.7free,5514.8used,224.0buff/cache;MiBSwap:98304.0total,98291.4free,12.6used.26753.1a
vailMem

anvefran@DESKTOP-B4IKBDH: ~/sistemasOp/proyecto2/proyecto
^vailMem
%Cpu(s):2.4us,4.8sy,0.0ni,92.4id,0.0wa,0.5hi,0.0si,0.0st;MiBMem:32398.5total,26637.2free,5537.3used,224.0buff/cache;MiBSwap:98304.0total,98291.4free,12.6used.26730.6a
vailMem
^
anvefran@DESKTOP-B4IKBDH:~/sistemasOp/proyecto2/proyecto$ ./subscriber 127.0.0.1 8090
Conectado exitosamente a 127.0.0.1 en el puerto 8090.
Escriba la metrica que desea recibir, cada subnivel incluirlo con un /
Metricas:
(Host#) Recibir toda la informacion del sistema
(CPU) Informacion de la CPU
(RAM) Informacion de la memoria RAM
(SWA) Informacion de la memoria SWAP
(/+/-) Wildcard de un nivel
(/#) Wildcard de varios niveles

ingrese el host y la metrica divididos por / >>host1/CPU/#
%Cpu(s):0.1us,1.8sy,0.0ni,97.3id,0.0wa,0.7hi,0.0si,0.0st
```

Funciona de la misma forma con host1/RAM o host1/RAM/# y host1/SWA o host1/SWA/# como se muestra en el código mencionado.

- Host1/CPU/idle o Host1/+/idle

```
anvefran@DESKTOP-B4IKBDH: ~/sistemasOp/proyecto2/proyecto
(CPU)      Informacion de la CPU
(RAM)      Informacion de la memoria RAM
(SWA)      Informacion de la memoria SWAP
(/+/)     Wildcard de un nivel
(/#)      Wildcard de varios niveles

ingrese el host y la metrica divididos por / >>host1/CPU/idle
99.3id
97.3id
98.1id
94.6id
99.2id
99.1id
98.7id
98.8id
92.1id
98.8id
91.2id
97.5id
97.4id

anvefran@DESKTOP-B4IKBDH: ~/sistemasOp/proyecto2/proyecto
anvefran@DESKTOP-B4IKBDH:~/sistemasOp/proyecto2/proyecto$ ./subscriber 127.0.0.1 8090
Conectado exitosamente a 127.0.0.1 en el puerto 8090.
Escoja la metrica que desea recibir, cada subnivel incluirlo con un /
Metricas:
(Host#)      Recibir toda la informacion del sistema
(CPU)        Informacion de la CPU
(RAM)        Informacion de la memoria RAM
(SWA)        Informacion de la memoria SWAP
(/+/)       Wildcard de un nivel
(/#)        Wildcard de varios niveles

ingrese el host y la metrica divididos por / >>host1/+/idle
99.1id
98.7id
98.8id
92.1id
98.8id
91.2id
97.5id

anvefran@DESKTOP-B4IKBDH: ~/sistemasOp/proyecto2/proyecto
anvefran@DESKTOP-B4IKBDH:~/sistemasOp/proyecto2/proyecto$ ^C
anvefran@DESKTOP-B4IKBDH:~/sistemasOp/proyecto2/proyecto$ ./publisher 127.0.0.1 8090
Conectado exitosamente a 127.0.0.1 en el puerto 8090.
Ingrese el Nombre de host: h1
%Cpu(s):0.6us,1.1sy,0.0ni,99.3id,0.0wa,0.0hi,0.0si,0.0st;MiBMem:32398.Stotal,26151.2free,6023.4used,224.0buff/cache;MiBSwap:98304.0total,98291.4free,12.6used.26244.5avallMem
%Cpu(s):0.0us,0.6sy,0.0ni,99.3id,0.0wa,0.1hi,0.0si,0.0st;MiBMem:32398.Stotal,26150.6free,6023.9used,224.0buff/cache;MiBSwap:98304.0total,98291.4free,12.6used.26244.0avallMem
%Cpu(s):0.8us,1.8sy,0.0ni,97.3id,0.0wa,0.1hi,0.0si,0.0st;MiBMem:32398.Stotal,26151.4free,6023.1used,224.0buff/cache;MiBSwap:98304.0total,98291.4free,12.6used.26244.8avallMem
%Cpu(s):0.3us,1.4sy,0.0ni,98.1id,0.0wa,0.2hi,0.0si,0.0st;MiBMem:32398.Stotal,26151.7free,6022.8used,224.0buff/cache;MiBSwap:98304.0total,98291.4free,12.6used.26245.1avallMem
%Cpu(s):1.9us,3.2sy,0.0ni,94.6id,0.0wa,0.2hi,0.0si,0.0st;MiBMem:32398.Stotal,26150.9free,6023.6used,224.0buff/cache;MiBSwap:98304.0total,98291.4free,12.6used.26244.3avallMem

anvefran@DESKTOP-B4IKBDH: ~/sistemasOp/proyecto2/proyecto
anvefran@DESKTOP-B4IKBDH:~/sistemasOp/proyecto2/proyecto$ ./subscriber 127.0.0.1 8090
Conectado exitosamente a 127.0.0.1 en el puerto 8090.
Escoja la metrica que desea recibir, cada subnivel incluirlo con un /
Metricas:
(Host#)      Recibir toda la informacion del sistema
(CPU)        Informacion de la CPU
(RAM)        Informacion de la memoria RAM
(SWA)        Informacion de la memoria SWAP
(/+/)       Wildcard de un nivel
(/#)        Wildcard de varios niveles

ingrese el host y la metrica divididos por / >>host1/+/idle
99.1id
98.7id
98.8id
92.1id
98.8id
91.2id
97.5id

anvefran@DESKTOP-B4IKBDH: ~/sistemasOp/proyecto2/proyecto
anvefran@DESKTOP-B4IKBDH:~/sistemasOp/proyecto2/proyecto$ ^C
anvefran@DESKTOP-B4IKBDH:~/sistemasOp/proyecto2/proyecto$ ./publisher 127.0.0.1 8090
Conectado exitosamente a 127.0.0.1 en el puerto 8090.
Ingrese el Nombre de host: h1
%Cpu(s):0.6us,1.1sy,0.0ni,99.3id,0.0wa,0.0hi,0.0si,0.0st;MiBMem:32398.Stotal,26151.2free,6023.4used,224.0buff/cache;MiBSwap:98304.0total,98291.4free,12.6used.26244.5avallMem
%Cpu(s):0.0us,0.6sy,0.0ni,99.3id,0.0wa,0.1hi,0.0si,0.0st;MiBMem:32398.Stotal,26150.6free,6023.9used,224.0buff/cache;MiBSwap:98304.0total,98291.4free,12.6used.26244.0avallMem
%Cpu(s):0.8us,1.8sy,0.0ni,97.3id,0.0wa,0.1hi,0.0si,0.0st;MiBMem:32398.Stotal,26151.4free,6023.1used,224.0buff/cache;MiBSwap:98304.0total,98291.4free,12.6used.26244.8avallMem
%Cpu(s):0.3us,1.4sy,0.0ni,98.1id,0.0wa,0.2hi,0.0si,0.0st;MiBMem:32398.Stotal,26151.7free,6022.8used,224.0buff/cache;MiBSwap:98304.0total,98291.4free,12.6used.26245.1avallMem
%Cpu(s):1.9us,3.2sy,0.0ni,94.6id,0.0wa,0.2hi,0.0si,0.0st;MiBMem:32398.Stotal,26150.9free,6023.6used,224.0buff/cache;MiBSwap:98304.0total,98291.4free,12.6used.26244.3avallMem
```

- Host1/RAM/free

```
anvefran@DESKTOP-B4IKBDH: ~/sistemasOp/proyecto2/proyecto
^C
anvefran@DESKTOP-B4IKBDH:~/sistemasOp/proyecto2/proyecto$ ./subscriber 127.0.0.1 8090
Conectado exitosamente a 127.0.0.1 en el puerto 8090.
Escoja la metrica que desea recibir, cada subnivel incluirlo con un /
Metricas:
(Host#)      Recibir toda la informacion del sistema
(CPU)        Informacion de la CPU
(RAM)        Informacion de la memoria RAM
(SWA)        Informacion de la memoria SWAP
(/+/)       Wildcard de un nivel
(/#)        Wildcard de varios niveles

ingrese el host y la metrica divididos por / >>host1/RAM/free
RAM free: 26120.6free
RAM free: 26121.5free
RAM free: 26126.0free

anvefran@DESKTOP-B4IKBDH: ~/sistemasOp/proyecto2/proyecto
anvefran@DESKTOP-B4IKBDH:~/sistemasOp/proyecto2/proyecto$ ^C
anvefran@DESKTOP-B4IKBDH:~/sistemasOp/proyecto2/proyecto$ ./publisher 127.0.0.1 8090
Conectado exitosamente a 127.0.0.1 en el puerto 8090.
Ingrese el Nombre de host: h1
%Cpu(s):1.0us,1.2sy,0.0ni,97.5id,0.0wa,0.3hi,0.0si,0.0st;MiBMem:32398.Stotal,26119.4free,6055.1used,224.0buff/cache;MiBSwap:98304.0total,98291.4free,12.6used.26212.8avallMem
%Cpu(s):0.1us,0.9sy,0.0ni,98.8id,0.0wa,0.2hi,0.0si,0.0st;MiBMem:32398.Stotal,26120.6free,6051.9used,224.0buff/cache;MiBSwap:98304.0total,98291.4free,12.6used.26214.0avallMem
%Cpu(s):0.7us,2.1sy,0.0ni,96.8id,0.0wa,0.3hi,0.0si,0.0st;MiBMem:32398.Stotal,26121.5free,6051.0used,224.0buff/cache;MiBSwap:98304.0total,98291.4free,12.6used.26214.9avallMem
%Cpu(s):0.4us,1.0sy,0.0ni,98.4id,0.0wa,0.2hi,0.0si,0.0st;MiBMem:32398.Stotal,26126.0free,6048.5used,224.0buff/cache;MiBSwap:98304.0total,98291.4free,12.6used.26219.4avallMem
%Cpu(s):0.2us,0.6sy,0.0ni,99.2id,0.0wa,0.0hi,0.0si,0.0st;MiBMem:32398.Stotal,26124.9free,6049.6used,224.0buff/cache;MiBSwap:98304.0total,98291.4free,12.6used.26218.3avallMem

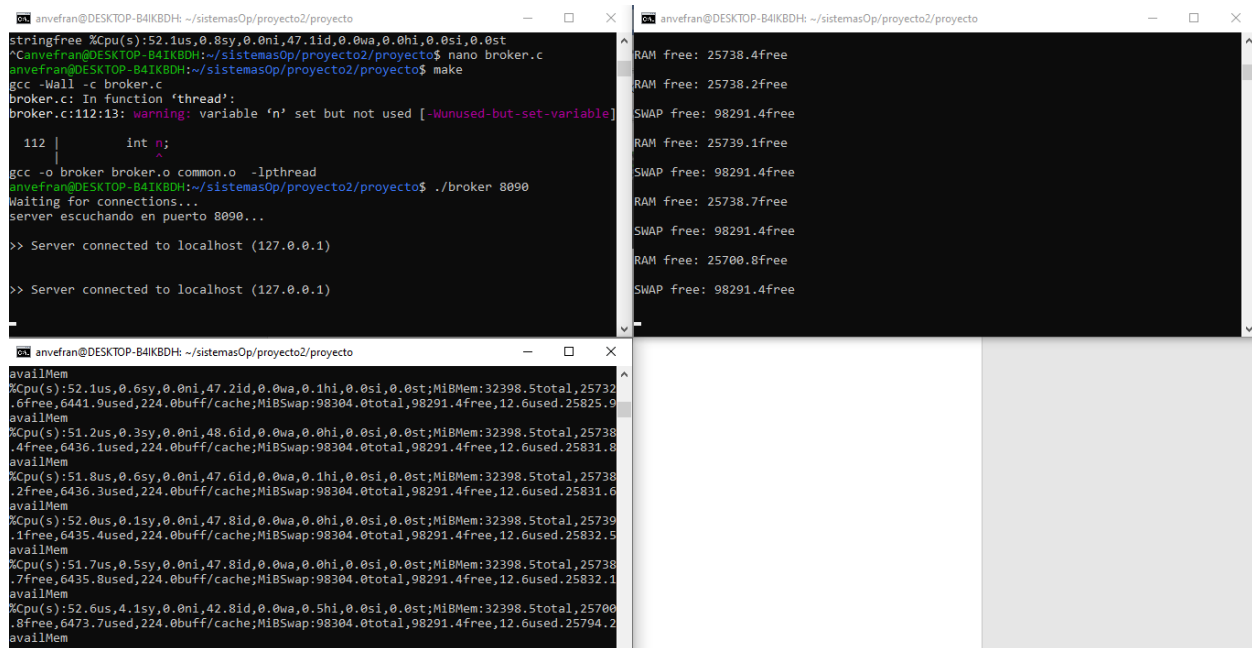
anvefran@DESKTOP-B4IKBDH: ~/sistemasOp/proyecto2/proyecto
anvefran@DESKTOP-B4IKBDH:~/sistemasOp/proyecto2/proyecto$ nano broker.c
broker.c: In function 'thread':
broker.c:112:13: warning: variable 'n' set but not used [-Wunused-but-set-variable]
    112 |         int n;
        |         ^~
gcc -o broker broker.o common.o -lpthread
anvefran@DESKTOP-B4IKBDH:~/sistemasOp/proyecto2/proyecto$ ./subscriber 127.0.0.1 8090
Conectado exitosamente a 127.0.0.1 en el puerto 8090.
Escoja la metrica que desea recibir, cada subnivel incluirlo con un /
Metricas:
(Host#)      Recibir toda la informacion del sistema
(CPU)        Informacion de la CPU
(RAM)        Informacion de la memoria RAM
(SWA)        Informacion de la memoria SWAP
(/+/)       Wildcard de un nivel
(/#)        Wildcard de varios niveles

ingrese el host y la metrica divididos por / >>
RAM free: 26120.6free
RAM free: 26121.5free
RAM free: 26126.0free

anvefran@DESKTOP-B4IKBDH: ~/sistemasOp/proyecto2/proyecto
anvefran@DESKTOP-B4IKBDH:~/sistemasOp/proyecto2/proyecto$ nano broker.c
broker.c: In function 'thread':
broker.c:112:13: warning: variable 'n' set but not used [-Wunused-but-set-variable]
    112 |         int n;
        |         ^~
gcc -o broker broker.o common.o -lpthread
anvefran@DESKTOP-B4IKBDH:~/sistemasOp/proyecto2/proyecto$ ./broker 8090
Server escuchando en puerto 8090...
>> Server connected to localhost (127.0.0.1)
>> Server connected to localhost (127.0.0.1)
>> Server connected to localhost (127.0.0.1)
```

Funciona de la misma forma para SWAP/free.

- Host1/+/free



```
anvefran@DESKTOP-B4IKBDH: ~/sistemasOp/proyecto2/proyecto
stringfree %Cpu(s):52.1us,0.8sy,0.0ni,47.1id,0.0wa,0.0hi,0.0si,0.0st
^C
anvefran@DESKTOP-B4IKBDH:~/sistemasOp/proyecto2/proyecto$ nano broker.c
anvefran@DESKTOP-B4IKBDH:~/sistemasOp/proyecto2/proyecto$ make
gcc -Wall -c broker.c
broker.c: In function 'tthread':
broker.c:112:13: warning: variable 'n' set but not used [-Wunused-but-set-variable]
    112 |         int n;
        |             ^
anvefran@DESKTOP-B4IKBDH:~/sistemasOp/proyecto2/proyecto$ ./broker 8090
Waiting for connections...
server escuchando en puerto 8090...

>> Server connected to localhost (127.0.0.1)

>> Server connected to localhost (127.0.0.1)

anvefran@DESKTOP-B4IKBDH: ~/sistemasOp/proyecto2/proyecto
availMem
%Cpu(s):52.1us,0.6sy,0.0ni,47.2id,0.0wa,0.1hi,0.0si,0.0st;MiBMem:32398.5total,25732.6free,6441.9used,224.0buff/cache;MiBSwap:98304.0total,98291.4free,12.6used.25825.9
availMem
%Cpu(s):51.2us,0.3sy,0.0ni,48.6id,0.0wa,0.0hi,0.0si,0.0st;MiBMem:32398.5total,25738.4free,6436.1used,224.0buff/cache;MiBSwap:98304.0total,98291.4free,12.6used.25831.8
availMem
%Cpu(s):51.8us,0.6sy,0.0ni,47.6id,0.0wa,0.1hi,0.0si,0.0st;MiBMem:32398.5total,25738.2free,6436.3used,224.0buff/cache;MiBSwap:98304.0total,98291.4free,12.6used.25831.6
availMem
%Cpu(s):52.0us,0.1sy,0.0ni,47.8id,0.0wa,0.0hi,0.0si,0.0st;MiBMem:32398.5total,25739.1free,6435.4used,224.0buff/cache;MiBSwap:98304.0total,98291.4free,12.6used.25832.5
availMem
%Cpu(s):51.7us,0.5sy,0.0ni,47.8id,0.0wa,0.0hi,0.0si,0.0st;MiBMem:32398.5total,25738.7free,6435.8used,224.0buff/cache;MiBSwap:98304.0total,98291.4free,12.6used.25832.1
availMem
%Cpu(s):52.6us,4.1sy,0.0ni,42.8id,0.0wa,0.5hi,0.0si,0.0st;MiBMem:32398.5total,25700.8free,6473.7used,224.0buff/cache;MiBSwap:98304.0total,98291.4free,12.6used.25794.2
availMem
```

5. Referencias

Sockets en C de Unix/Linux. (2022). Chuidiang.org.

[https://www.chuidiang.org/clinux/sockets/sockets_simp.php#:~:text=LOS%20SOCKETS,](https://www.chuidiang.org/clinux/sockets/sockets_simp.php#:~:text=LOS%20SOCKETS)

-

Una forma de Desde el punto de vista, write() del lenguaje C.

rafariva-classroom. (2022, January 6). *rafariva-classroom/taller07-anvefran: taller07-anvefran created by GitHub Classroom.* GitHub. <https://github.com/rafariva-classroom/taller07-anvefran>

parzibyte. (2018, November 13). *Separar cadena a partir de delimitadores en C con strtok* - Parzibyte's blog. Parzibyte's Blog. <https://parzibyte.me/blog/2018/11/13/separar-cadena-delimitadores-c-strtok/>

c - *Reading from stdin*. (n.d.). Stack Overflow. Retrieved July 18, 2022, from <https://stackoverflow.com/questions/15883568/reading-from-stdin>

(2022). Site24x7.com. <https://support.site24x7.com/portal/en/kb/articles/how-is-cpu-utilization-calculated-for-a-linux-server-monitor>

baeldung. (2021, August 25). *Get Overall CPU Usage on Linux | Baeldung on Linux*. Baeldung on Linux. <https://www.baeldung.com/linux/get-cpu-usage>

Ejecución de procesos - Title. (2022). Ulgc.es. http://labsopa.dis.ulpgc.es/prog_c/PROCES.HTM

afontelasanchez. (2018, June 26). *Raiola Networks es tu hosting con soporte 24/7*. Raiola Networks. <https://raiolanetworks.es/blog/memoria-ram-usada-memoria-ram-libre-linux/>

El comando Free | rm-rf.es. (2008, August 23). # Rm-Rf.es | Blog de Un Sysadmin Unix, GNU/Linux, Windows Y Lo Que Haga Falta... <https://rm-rf.es/el-comando-free/#:~:text=El%20comando%20Free%20en%20Linux,buffer%20consumida%20por%20el%20Kernel.>

Junior, R. (2021, August 17). *What is difference between free and available memory in Linux?* - Techtrix. Techtrix. <https://techtrix.co/what-is-difference-between-free-and-available-memory-in-linux/>

Poppyto. (2022). *Convert float to char in C | Convert Data Types*. Convertdatatype.com. <https://www.convertdatatype.com/Convert-float-to-char-in-C.html>

Frank. (2009, November 17). *how to store printf into a variable?* Stack Overflow. <https://stackoverflow.com/questions/1745726/how-to-store-printf-into-a-variable>

pthread_detach(3) - *Linux manual page*. (2021). Man7.org. https://man7.org/linux/man-pages/man3/pthread_detach.3.html