Manual técnico

Consideraciones

El proyecto fue realizada en:

- En Arch Linux
- 3Gb de memoria RAM
- 4Gb de memoria SWAP
- 75GB de almacenamiento interno
- Procesador intel core i5 7gen

Aplicación de Inotify para monitoreo de la raiz “/”:

Para poder utilizar inotify debemos de incluirla a nuestra aplicacion de esta manera:

```
#include <sys/inotify.h>
```

Cuando iniciamos inotify nos retorna un entero si el entero es menor que cero quiere decir que inotify no se inició. de la siguiente manera:

```
int fd;
fd = inotify_init();
   if(fd<0){
       perror("notify no se inicio");
   }
```

Para indicar la ruta en donde queremos estar observando los cambios haremos uso de inotify_add_watch el cual también, el cual recibe tres argumentos, el primero es el estado de iniciación de inotify, el segundo es la ruta a la cual queremos visualizar, y el tercer argumento es la lista de acciones que queremos monitorizar de la siguiente manera, si inotify_add_watch devuelve -1 quiere decir que no se pudo iniciar la monitorizacion.

```
int watch_desc;
watch_desc= inotify_add_watch(fd,"/",IN_CREATE|IN_MODIFY|IN_DELETE);
   if(watch_desc==-1){
       printf("no se pudo agreagar el visor al forder /home/wilson");
   }else{
       printf("Monitoreando /");
   }
```

como se observa el tercer argumento se describe a continuación:

- IN_CREATE: Para monitorizar la creación de archivos.
- IN_MODIFY: Para monitorizar la modificación de archivos.
- IN_DELETE: Para monitorizar la eliminación de archivos.

Para que nuestro programa se quede escuchando indefinidamente se hace uso de un ciclo infinito, en este caso un while(1)

```
While(1){...}
```

Ahora bien estaremos escuchando por cambios y ver el estado de la siguiente manera

```
int total_read=read(fd,buffer,BUFFER_LEN);
```

dónde total_read como su nombre lo indica almacenará, si total_read llega a ser menor a cero quiere decir que la lectura tuvo un error y lo manejamos de la siguiente manera.

```
if(total_read<0){
            perror("error de lectura");
       }
```

ahora para llevar el control de las lecturas y ver si hay alguna lectrua nueva nos ayudaremos de un contador al cual llamaremos “i”, si i no coincide con el total_read y total read es mayor que i y total_read es mayor a cero entonces quiere decir que hay un cambio en el directorio de la siguiente manera:

```
int i=0;
while(i<total_read){...}
```

dentro de este while se captura el tipo de lectura y se imprime el mensaje necesario, para esto haremos uso de la estructura inotify_event, la cual capturará el evento de la siguiente manera:

```
struct inotify_event *event=(struct inotify_event*)&buffer[i];
```

si el evento en este caso event tiene longitud quiere decir que se capturo de buena manera el evento y se procede a escribirlo

```
if(event->len){...}
```

Como se muestra en la imagen si se captura un evento IN_CREATE y luego se verifica si es en un directorio o en un archivo lo mismo se hace para IN_MODIFY y para IN_DELETE. Luego aumentamos i agregando el tamaño del evento leido y del evento de monitoreo.

```
if(event->len){
                if(event->mask & IN_CREATE){
                    if(event->mask & IN_ISDIR){
                        printf("Directorio %s fue creado\n",event->name);
                    }else{
                        printf("Archivo %s fue creado\n",event->name);
                    }
                }
                if(event->mask & IN_MODIFY){
                    if(event->mask & IN_ISDIR){
                        printf("Directorio %s fue modificado\n",event->name);
                    }else{
                        printf("Archivo %s fue modificado\n",event->name);
                    }
                }
                if(event->mask & IN_DELETE){
                    if(event->mask & IN_ISDIR){
                        printf("Directorio %s fue eliminado\n",event->name);
                    }else{
                        printf("Archivo %s fue eliminado\n",event->name);
                    }
                }
                i+=MONITOR_EVENT_SIZE+event->len;
            }
```