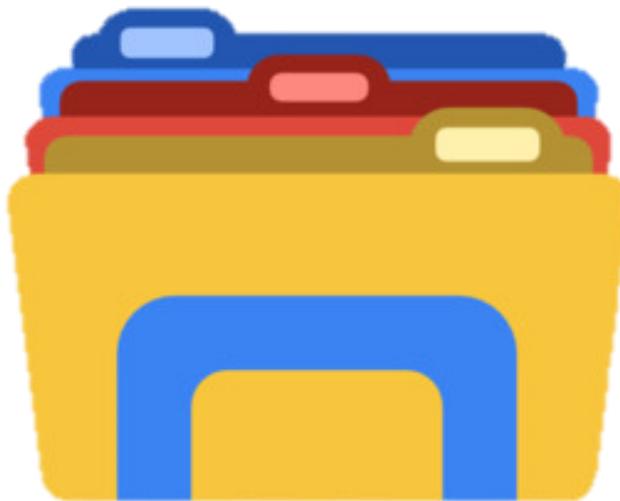


Universidad Autónoma de Aguascalientes

CENTRO DE CIENCIAS BÁSICAS

ING. EN SISTEMAS COMPUTACIONALES

DOCUMENTACIÓN: ADMINISTRADOR DE ARCHIVOS



Alumnos:

- Francisco de Jesús Méndez Lara
- Emmanuel Muñoz Cerdá
- Sandra Olimpia Estrella Prieto García
- Gerardo Femat Delgado

Profesor: Javier Santiago Cortes López

Fecha: 22/06/22

4^{to} SEMESTRE, GRUPO C

Introducción: En este proyecto se realizará un administrador de archivos desarrollado en Shell Script. Mediante el uso de una interfaz de texto para el usuario (TUI). Usando como medio para su creación y programación una máquina de Virtual Box y Visual Studio Code.

Marco teórico:

- ❖ **Tecnología:** Se hizo uso de Linux Mint como sistema operativo, para estar programando los Shells. El tipo de shell que utilizado fue bash.
- ❖ **Dialog:** Es un comando de GNU/linux que permite crear cuadros de dialogo en el terminal para que se utilice en los scripts. Suele ser muy usado para administración de sistemas, instalación de distribuciones, configuración de aplicaciones, etc. También permite el control de programas en un tiempo de ejecución.
- ❖ **ncurses:** Son librerías que provee una API que permite al programador escribir interfaces basadas en texto y TUI's. También optimiza el refresco de la pantalla, lo que permite reducir la latencia experimentada cuando se usan intérpretes de comandos remotos. Estas librerías ayudan a trabajar con funciones similares a las contenidas en la librería conio.h del viejo turboC de Borland, entre las cuales están: gotoxy(), clrscr(), textColor(), getch(), etc. Con ncurses se pueden hacer aplicaciones gráficas elegantes en modo texto y de forma muy fácil para ser utilizadas en terminal.

Temas del curso que se usaron a lo largo del proyecto:

- **Shell Script:** Es un grupo de comandos, funciones y variables, tienen la misma sintaxis de varios lenguajes de programación y está fuertemente influenciado por el lenguaje C. Los Shell Scripts son capaces de manejar las mismas tareas que estos. Los scripts deben escribirse en un archivo de texto al que se le otorga permisos de ejecución. Además, existen dos tipos de scripts: los basados en bourne shell y los derivados de C.

Ya que se definió lo que es un Shell Script, ahora se definirá lo que es ***Shell Scripting***. Esta es la técnica de diseñar y crear un Script mediante un Shell de un Sistema Operativo, o un editor de texto. Este siendo un tipo de lenguaje de programación que generalmente es interpretado durante la ejecución.

- **Metacaracteres:** El metacarácter son caracteres específicos que interpreta el sistema operativo mandando indicaciones a la computadora cómo proceder. Nuestro sistema operativo leerá un metacarácter e interpretará lo que significa, para posteriormente realizar una acción basada en esa interpretación. Tal acción puede incluir instruir al sistema para que separe las declaraciones o buscar diferentes deletreos de una palabra, así como cambiar los colores e inclusive imprimir cuadros.

Desarrollo de los puntos vistos a lo largo del proyecto:

Se desarrollaron los puntos solicitados en el proyecto, iniciando una investigación sobre “*midnight commander*”, una vez observado su funcionamiento, se prosiguió a seguir investigando, pero enfocado a lo que podría servir para crear una interfaz de texto. Esto fue debido a que se consideró más importante aprender el uso del programa para realizar la interfaz ya que no se vieron durante clases, que los scripts mismos. Investigando, se encontró una paquetería llamada “*dialog*”, la cual solucionaba gran parte de los problemas planteados con la interfaz, ya que cuenta con herramientas de gran utilidad para configurar el administrador de archivos.

Se comenzó por programar los scripts de menú, tras esto se fue conectando la interfaz para que estuviera ciclada, mediante la etiqueta “*--fselect*” se abre el menú para seleccionar archivos o directorios al iniciar a trabajar con los scripts, ya que esa etiqueta retorna la ruta en la que el usuario esta seleccionando. Se programaron las condiciones para detectar si el usuario está trabajando con un directorio o un archivo dentro de un menú de opciones, esto provoca que se le muestren diferentes opciones: si es un directorio o si es un archivo. Ya teniendo las diferentes opciones, a cada opción se le dio su script correspondiente, de los cuales se desarrollaron 3

tipos: scripts para directorios, para archivos y para ambos (estos los llamamos generales).

Retomando lo mencionado, el “*--fselect*” proporciona la ruta donde se encuentra el usuario, esto se pasa por parámetro a los scripts y de ahí se valida si es una ruta aceptable, si el usuario tiene permisos para hacer las modificaciones o tratar el directorio del archivo como es debido. Después se investigó la manera en la que se podría ejecutar el script al iniciar sesión en la máquina virtual de ejemplo, para que cada vez que se haga un login se ejecute en pantalla completa. Para lograrlo se usó Fedora, configurándolo para que al arrancar el SO el usuario realice su login e inicie el script de forma automática.

También se programaron las funciones extendidas, estas funcionan tanto para archivos como directorios: cambio de permisos, cambio de propietario y cambio de grupo, únicamente para directorios: agregar prefijos. Para cambiar los permisos, se necesita hacer log in como root o el usuario al que le pertenece ese archivo, para modificar los permisos del grupo, lo mismo se puede hacer desde el usuario root o el usuario al que le pertenece ese archivo. Para cambiar el propietario de una archivo o directorio, se debe hacer solamente accediendo con root. Para el prefijo se debe seleccionar una carpeta a la que su contenido se le desea añadir los prefijos.

Finalmente se desarrolló el makefile, el cual revisa que estén los programas necesarios, todos los scripts y cambia a modo de ejecución cada uno de los scripts para que el usuario pueda ejecutar el main.sh.

Problemas encontrados a la hora de desarrollar el proyecto y sus soluciones:

La principal dificultad que se presento fue el saber cómo configurar el dialog, ya que se estuvo leyendo constantemente su manual e investigando etiquetas específicas de esté en internet, pero esto último no fue de mucha ayuda, ya que en su mayoría son ejemplos muy básicos que no cumplía el objetivo, por lo tanto fue batallar a base de prueba y error, hasta que se logrará comprender el funcionamiento de este, una vez comprendido, generar lo demás se volvió bastante sencillo.

Conclusión: El Shell script es una manera de generar programas de administración muy sencillos, esté proyecto sin duda fue un reto, ya sea por la dificultad que se presentaba de forma lógica al momento de crear el administrador de archivos desde cero, así como la poca información que se encontraba a la hora de investigar. Se tuvieron que realizar investigaciones más profundas o exactas, para guiarnos e incluso a veces ni así se lograba encontrar algo. En gran parte del proyecto, como se mencionó con anterioridad en el desarrollo de este documento, el administrador de archivos fue creado a base de prueba y error, con el fin de aprender el funcionamiento de algunos programas como el dialog o comprender el funcionamiento de los comandos que utilizamos para programar los demás scipts.

Conclusiones por integrante:

Gerardo Femat Delgado: Durante la creación del proyecto reforzamos los conocimientos obtenidos en el curso, tomando conciencia de las capacidades que tiene el shell scripting y su gran importancia en el entorno para administración y mantenimiento de los sistemas. El mayor reto sin ninguna duda fue investigar y comprender el funcionamiento de “dialog” ya que fue muy complicado comprender como retornaba aquello que seleccionaba el usuario y si seleccionaba un botón específico. Una vez comprendido qué, cómo y por qué retornaba valores concretos todo fue más sencillo. Por igual debí investigar sobre las funciones en bash, para que fuera fácil el manejo de la interfaz. Una vez con los conocimientos básicos solo se tuvieron que crear los scipts conociendo como se retornaban los valores y manejarlos con readlink y dirname, comandos muy útiles al manejar archivos o directorios. En cuanto a los scripts a pesar de ser muchos fueron sencillos de realizar, con lo visto en clases y un poco de investigación fue suficiente. El proyecto en un contexto general me gusto hacerlo, el shell scripting es una herramienta muy poderosa en los sistemas like-UNIX y sobre todo imagino en servidores.

Francisco de Jesús Méndez Lara: Durante el tiempo que estuve investigando sobre la problemática planteada a simple vista no parecía un gran reto, ya que con lo aprendido en clases sobre lo que es el Shell Scripting sonaba bastante sencillo, las cosas se tornaron complicadas cuando comenzamos a investigar sobre la interfaz de texto, ya que no había mucha información en internet y la que había era bastante sencilla, tratamos de analizar lo que era el midnight commander, pero al ver fue desarrollada casi en su totalidad en C, decidimos dejar de investigar su código para no cerrarnos a no intentarlo solamente con Shell Scripts. Al investigar sobre dialog también se nos presentó diferentes obstáculos, ya que no terminábamos de comprender como funcionaba, si retornaba valores, que modificaciones hacía, pero al menos yo después de leer todo el manual de usuario brindado por el “man” vi diferentes tipos de ventanas que nos proveía, por lo que ya teníamos por donde comenzar. Fue bastante entretenido experimentar con el dialog, lo cual nos dio un parámetro muy amplio para comenzar a hacer la TUI, donde finalmente adaptar los Scripts para que se armonizara con nuestra interfaz fuera bastante buena, la verdad la capacidad de configuración que tienen los Shell scripts, la verdad me sorprendió, entendí mejor el porque es muy usado para mantenimiento y administración de sistemas.

Sandra Olimpia Estrella Prieto García: A lo largo de este proyecto, me di cuenta de que es de suma importancia el saber investigar bien tus fuentes de consulta, para poder encontrar algo que te guíe u oriente bien. No fue nada fácil el lograr avanzar el proyecto, ya que a pesar de que teníamos las bases para poder estar programando algunos scripts, el saber cómo hacer uso de otras cosas como el “dialog”, fue el que nos detuvo. Ya que tuvo su cierto grado de complejidad el saber cómo funciona y usarlo.

Emmanuel Muñoz Cerda: El desarrollo del proyecto me hizo darme cuenta de lo poderosos que pueden ser los shell scripts, es increíble todos los procesos que se pueden automatizar usando solo shell scripts. Es nuevo e impactante el poder que tienen las variables dentro de un shell script, donde a diferencia de los lenguajes de programación que se han visto a lo largo de la carrera (C/C++ y Java), estas variables pueden almacenar una infinidad de resultados, además de que no es necesario indicar el tipo de dato que van a almacenar, así como tampoco es necesario declararlas antes de usarlas, se puede declarar una variable en el momento en el que se va a usar. Y usar posteriormente a lo largo del script. Además de todo lo anterior, me di cuenta de que lo que le da poder a un shell script es la posibilidad de usar comandos de la terminal para realizar diversas operaciones.

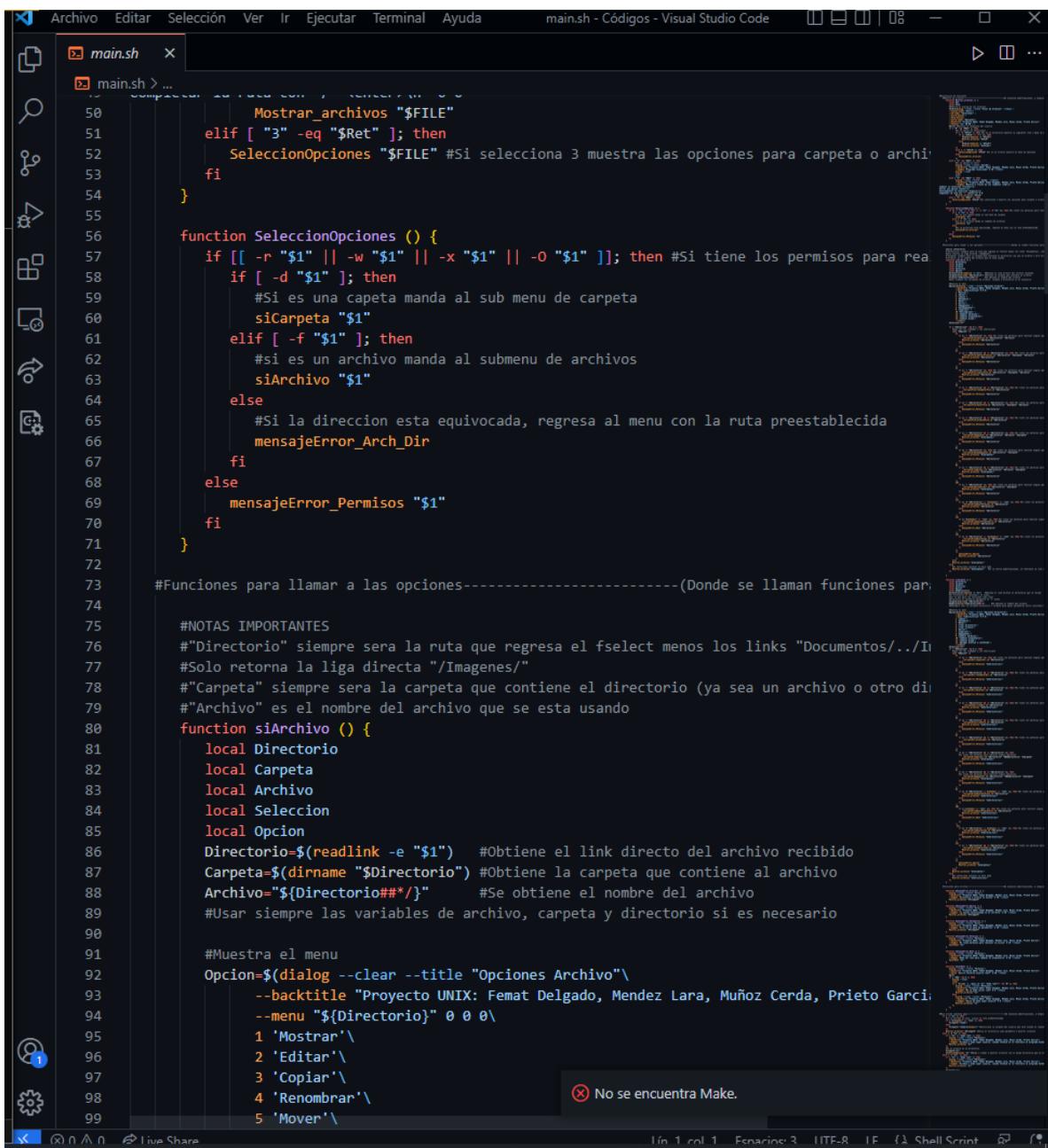
Anexos:

Link para ver todos los códigos de proyecto: https://eduuaa-my.sharepoint.com/:f/g/personal/al289539_edu_uaa_mx/EIF2BKQ2UjxPtnmDT3BKTl8BGOEGpY9SWXdnfPFzPPV85A?e=PIuUM1

Link con todo el proyecto: https://eduuaa-my.sharepoint.com/:f/g/personal/al289539_edu_uaa_mx/EmWUfBnZ3oNBmO0LOwU9apiBB-2cyYuzZsAzWIx2SaTYVw?e=LNf4V7

Función de cada código:

- **Main:** Script principal del programa, lugar donde se cicla el fselector y todos los scripts que conforman el proyecto.

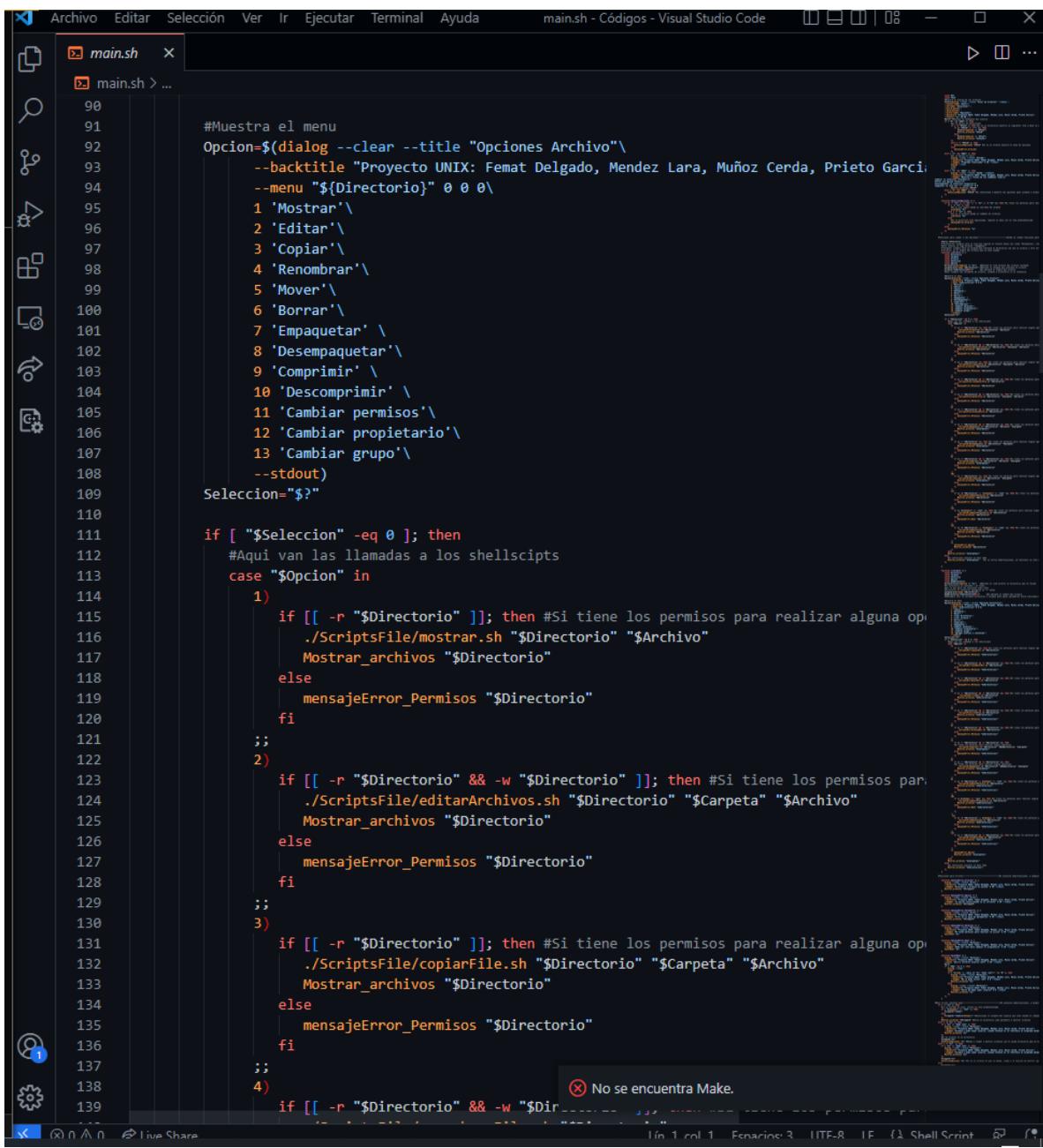


main.sh - Códigos - Visual Studio Code

```
main.sh x
main.sh > ...
50     Mostrar_archivos "$FILE"
51 elif [ "3" -eq "$Ret" ]; then
52     SeleccionOpciones "$FILE" #Si selecciona 3 muestra las opciones para carpeta o archivo
53 fi
54 }
55
56 function SeleccionOpciones () {
57 if [[ -r "$1" || -w "$1" || -x "$1" || -O "$1" ]]; then #Si tiene los permisos para rea
58     if [ -d "$1" ]; then
59         #Si es una carpeta manda al sub menu de carpetas
60         siCarpeta "$1"
61     elif [ -f "$1" ]; then
62         #Si es un archivo manda al submenu de archivos
63         siArchivo "$1"
64     else
65         #Si la direccion esta equivocada, regresa al menu con la ruta pre establecida
66         mensajeError_Arch_Dir
67     fi
68 else
69     mensajeError_Permisos "$1"
70 fi
71 }
72
73 #Funciones para llamar a las opciones----- (Donde se llaman funciones para
74
75 #NOTAS IMPORTANTES
76 # "Directorio" siempre sera la ruta que regresa el fselect menos los links "Documentos/../
77 # Solo retorna la liga directa "/Imagenes/"
78 # "Carpeta" siempre sera la carpeta que contiene el directorio (ya sea un archivo o otro di
79 # "Archivo" es el nombre del archivo que se esta usando
80 function siArchivo () {
81 local Directorio
82 local Carpeta
83 local Archivo
84 local Seleccion
85 local Opcion
86 Directorio=$(readlink -e "$1") #Obtiene el link directo del archivo recibido
87 Carpeta=${dirname "$Directorio"} #Obtiene la carpeta que contiene al archivo
88 Archivo="${Directorio##*/}" #Se obtiene el nombre del archivo
89 #Usar siempre las variables de archivo, carpeta y directorio si es necesario
90
91 #Muestra el menu
92 Opcion=$(dialog --clear --title "Opciones Archivo" \
93             --backtitle "Proyecto UNIX: Femat Delgado, Mendez Lara, Muñoz Cerda, Prieto Garcí
94             --menu "${Directorio}" 0 0 \
95             1 'Mostrar'\ \
96             2 'Editar'\ \
97             3 'Copiar'\ \
98             4 'Renombrar'\ \
99             5 'Mover'\ \

```

No se encuentra Make.



main.sh - Códigos - Visual Studio Code

```
90
91     #Muestra el menu
92     Opcion=$(dialog --clear --title "Opciones Archivo" \
93         --backtitle "Proyecto UNIX: Femat Delgado, Mendez Lara, Muñoz Cerda, Prieto Garcia" \
94         --menu "${Directorio}" 0 0 0 \
95         1 'Mostrar' \
96         2 'Editar' \
97         3 'Copiar' \
98         4 'Renombrar' \
99         5 'Mover' \
100        6 'Borrar' \
101        7 'Empaquetar' \
102        8 'Desempaquetar' \
103        9 'Comprimir' \
104        10 'Descomprimir' \
105        11 'Cambiar permisos' \
106        12 'Cambiar propietario' \
107        13 'Cambiar grupo' \
108        --stdout)
109     Seleccion="$?"
110
111 if [ "$Seleccion" -eq 0 ]; then
112     #Aqui van las llamadas a los shellscripts
113     case "$Opcion" in
114         1)
115             if [[ -r "$Directorio" ]]; then #Si tiene los permisos para realizar alguna op
116                 ./ScriptsFile/mostrar.sh "$Directorio" "$Archivo"
117                 Mostrar_archivos "$Directorio"
118             else
119                 mensajeError_Permisos "$Directorio"
120             fi
121         ;;
122         2)
123             if [[ -r "$Directorio" && -w "$Directorio" ]]; then #Si tiene los permisos par
124                 ./ScriptsFile/editarArchivos.sh "$Directorio" "$Carpeta" "$Archivo"
125                 Mostrar_archivos "$Directorio"
126             else
127                 mensajeError_Permisos "$Directorio"
128             fi
129         ;;
130         3)
131             if [[ -r "$Directorio" ]]; then #Si tiene los permisos para realizar alguna op
132                 ./ScriptsFile/copiarFile.sh "$Directorio" "$Carpeta" "$Archivo"
133                 Mostrar_archivos "$Directorio"
134             else
135                 mensajeError_Permisos "$Directorio"
136             fi
137         ;;
138         4)
139             if [[ -r "$Directorio" && -w "$Dir
```

main.sh - Códigos - Visual Studio Code

```
main.sh > ...
151 ;;
152 ;;
6) if [[ -r "$Directorio" && -w "$Directorio" ]]; then #Si tiene los permisos para
   ./ScriptsFile/eliminaFile.sh "$Directorio"
else
   mensajeError_Permisos "$Directorio"
fi
;;
7) if [[ -r "$Directorio" && -w "$Directorio" ]]; then #Si tiene los permisos para
   ./ScriptsG/empaquetar.sh "$Directorio" "$Archivo" "$Carpeta"
   Mostrar_archivos "${Carpeta}/"
else
   mensajeError_Permisos "$Directorio"
fi
;;
8) if [[ -r "$Directorio" ]]; then #Si tiene los permisos para realizar alguna op
   ./ScriptsG/desempaquetar.sh "$Directorio" "$Carpeta"
   Mostrar_archivos "${Carpeta}/"
else
   mensajeError_Permisos "$Directorio"
fi
;;
9) if [[ -r "$Directorio" && -w "$Directorio" ]]; then #Si tiene los permisos para
   ./ScriptsG/comprimir.sh "$Directorio" "$Archivo" "$Carpeta"
   Mostrar_archivos "${Carpeta}/"
else
   mensajeError_Permisos "$Directorio"
fi
;;
10) if [[ -r "$Directorio" ]]; then #Si tiene los permisos para realizar alguna op
   ./ScriptsG/descomprimir.sh "$Directorio" "$Carpeta"
   Mostrar_archivos "${Carpeta}/"
else
   mensajeError_Permisos "$Directorio"
fi
;;
11) if [[ -o "$Directorio" || "$(whoami)" == 'root' ]]; then #Si tiene los permiso
   ./ScriptsG/cambiarPermisos.sh "$Directorio"
   Mostrar_archivos "$Directorio"
else
   mensajeError_Permisos "$Directorio"
fi
;;
12) if [[ "$(whoami)" == 'root' ]]; then
   /ScriptsG/cambiarDronietario.sh "$Directorio"
```

No se encuentra Make.

The screenshot shows a Visual Studio Code interface with a terminal window open. The terminal output is as follows:

```
No se encuentra Make.
```

The main code editor window displays a shell script named `main.sh`. The script contains several functions and conditional statements. A tooltip in the bottom right corner of the code editor says "No se encuentra Make." The code itself includes comments in Spanish explaining its purpose, such as checking if the user is root or if they have permissions to change files in a directory. It also handles cases where a directory is selected and where modifications are made.

```
if [[ "$(whoami)" == 'root' ]]; then #Si tiene los permisos para realizar alguno de los siguientes cambios
    ./ScriptsG/cambiarPropietario.sh "$Directorio"
    Mostrar_archivos "$Directorio"
else
    mensajeError_Root "$Directorio"
fi
;;
13)
    if [[ -O "$Directorio" || "$(whoami)" == 'root' ]]; then #Si tiene los permisos para realizar alguno de los siguientes cambios
        ./ScriptsG/cambiarGrupo.sh "$Directorio"
        Mostrar_archivos "$Directorio"
    else
        mensajeError_Permisos "$Directorio"
    fi
;;
*)
    mensajeError_Opcion
    Mostrar_archivos "$Directorio"
;;
esac
Mostrar_archivos "${Carpeta}/"
else
    #Si selecciono cancelar no hace nada
    Mostrar_archivos "${Carpeta}/" #Si no sufrio modificaciones, se retornara la ruta anterior
fi
}

function siCarpeta () {
local Directorio
local Carpeta
local Seleccion
local Opcion
local NomDirectorio
Directorio=$(readlink -e "$1") #Obtiene el link directo al directorio que se recibe
#El directorio no contiene / al final,
#por lo que para las funciones como crear
#directorios es necesario agregarle el "/" antes
Carpeta=$(dirname "$Directorio")
NomDirectorio="${Directorio##*/}" #Se obtiene el nombre del archivo
#¡¡Siempre usar la variable directorio o carpeta para pasar parametros entre funciones!

#Muestra el menu
Opcion=$(dialog --clear --title "Opciones Directorio" \
--backtitle "Proyecto UNIX: Femat Delgado, Mendez Lara, Muñoz Cerdá, Prieto García" \
--menu "${Directorio}" 0 0 0 \
1 'Copiar'\` \
2 'Renombrar'\` \
3 'Mover'\` \
4 'Crear directorio'\`)
```

The screenshot shows a Visual Studio Code interface with a terminal window open on the right side. The terminal window displays a shell script named `main.sh`. The script contains several conditional blocks (if statements) that check for specific file permissions and execute corresponding shell scripts if they have the required permissions. A tooltip in the terminal window indicates that 'No se encuentra Make' (Makefile not found). The terminal also shows some command history and output from previous executions.

```
Archivo Editar Selección Ver Ir Ejecutar Terminal Ayuda main.sh - Códigos - Visual Studio Code
main.sh x
main.sh > ...
259 Seleccion="$?"
260 if [ "$Seleccion" -eq 0 ]; then
261     #Aqui van las llamadas a los shellscripts
262     case "$Opcion" in
263         1)
264             if [[ -r "$Directorio" ]]; then #Si tiene los permisos para realizar alguna operacion
265                 ./ScriptsDir/copiarDir.sh "$Directorio"
266             else
267                 mensajeError_Permisos "${Directorio}/"
268             fi
269         ;;
270         2)
271             if [[ -r "$Directorio" && -w "$Directorio" ]]; then #Si tiene los permisos para renombrar
272                 ./ScriptsDir/renombrarDir.sh "$Directorio"
273             else
274                 mensajeError_Permisos "${Directorio}/"
275             fi
276         ;;
277         3)
278             if [[ -r "$Directorio" && -w "$Directorio" ]]; then #Si tiene los permisos para mover
279                 ./ScriptsDir/moverDir.sh "$Directorio"
280             else
281                 mensajeError_Permisos "${Directorio}/"
282             fi
283         ;;
284         4)
285             if [[ -r "$Directorio" && -w "$Directorio" ]]; then #Si tiene los permisos para crear directorio
286                 ./ScriptsDir/creaDir.sh "$Directorio"
287                 Mostrar_archivos "${Directorio}/"
288             else
289                 mensajeError_Permisos "${Directorio}/"
290             fi
291         ;;
292         5)
293             if [[ -r "$Directorio" && -w "$Directorio" ]]; then #Si tiene los permisos para crear archivo
294                 ./ScriptsFile/creaFile.sh "$Directorio"
295                 Mostrar_archivos "${Directorio}/"
296             else
297                 mensajeError_Permisos "${Directorio}/"
298             fi
299         ;;
300         6)
301             if [[ -r "$Directorio" && -w "$Directorio" ]]; then #Si tiene los permisos para eliminar directorio
302                 ./ScriptsDir/eliminaDir.sh "$Directorio"
303             else
304                 mensajeError_Permisos "${Directorio}/"
305             fi
306         ;;
307         7)
308             if [[ -r "$Directorio" && -w "$Dir" ]]; then #Si tiene los permisos para realizar alguna operacion
309                 #<!-- tareas de ejecucion de comando -->
310             fi
311         ;;
312     esac
313     echo "Operacion realizada con exito"
314 fi
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
618
619
619
620
621
622
623
624
625
626
627
628
629
629
630
631
632
633
634
635
636
637
638
639
639
640
641
642
643
644
645
646
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
679
679
680
681
682
683
684
685
686
687
687
688
689
689
690
691
692
693
694
695
696
697
697
698
699
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
718
719
719
720
721
722
723
724
725
726
727
728
729
729
730
731
732
733
734
735
736
737
738
739
739
740
741
742
743
744
745
746
747
748
749
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
769
770
771
772
773
774
775
776
777
778
779
779
780
781
782
783
784
785
786
787
787
788
789
789
790
791
792
793
794
795
796
797
797
798
799
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
818
819
819
820
821
822
823
824
825
826
827
828
829
829
830
831
832
833
834
835
836
837
838
839
839
840
841
842
843
844
845
846
847
848
849
849
850
851
852
853
854
855
856
857
858
859
859
860
861
862
863
864
865
866
867
868
869
869
870
871
872
873
874
875
876
877
878
879
879
880
881
882
883
884
885
886
887
887
888
889
889
890
891
892
893
894
895
896
897
897
898
899
899
900
901
902
903
904
905
906
907
908
909
909
910
911
912
913
914
915
916
917
918
919
919
920
921
922
923
924
925
926
927
928
929
929
930
931
932
933
934
935
936
937
938
939
939
940
941
942
943
944
945
946
947
948
949
949
950
951
952
953
954
955
956
957
958
959
959
960
961
962
963
964
965
966
967
968
969
969
970
971
972
973
974
975
976
977
978
979
979
980
981
982
983
984
985
986
987
987
988
989
989
990
991
992
993
994
995
996
997
998
999
999
1000
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1079
1080
1081
1082
1083
1084
1085
1086
1087
1087
1088
1089
1089
1090
1091
1092
1093
1094
1095
1095
1096
1097
1098
1099
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1188
1189
1190
1191
1192
1193
1194
1195
1195
1196
1197
1198
1199
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1297
1298
1299
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1397
1398
1399
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1497
1498
1499
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1597
1598
1599
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1697
1698
1699
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1797
1798
1799
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1897
1898
1899
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1988
1989
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2088
2089
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2188
2189
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2198
2199
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2288
2289
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2359
2360
2361
2362
```



```
main.sh - Códigos - Visual Studio Code
```

```
main.sh > ...

369 #Funciones para errores----- (No necesita modificaciones, a excepción de las que se mencionan)
370
371 function mensajeError_Arch_Dir () {
372     dialog --clear --title 'Error' \
373         --backtitle "Proyecto UNIX: Femat Delgado, Mendez Lara, Muñoz Cerda, Prieto Garcia" \
374         --msgbox 'Directorio o archivo no existe' 6 30 --stdout
375     Mostrar_archivos "$Filepath"
376 }
377
378
379 function mensajeError_Opcion () {
380     dialog --clear --title 'Error' \
381         --backtitle "Proyecto UNIX: Femat Delgado, Mendez Lara, Muñoz Cerda, Prieto Garcia" \
382         --msgbox 'La opcion seleccionada no es correcta' 6 30 --stdout
383     Mostrar_archivos "$Filepath"
384 }
385
386 function mensajeError_Parametros () {
387     dialog --clear --title 'Error' \
388         --backtitle "Proyecto UNIX: Femat Delgado, Mendez Lara, Muñoz Cerda, Prieto Garcia" \
389         --msgbox 'Error en el paso de parametros' 6 30 --stdout
390     Mostrar_archivos "$Filepath"
391 }
392
393 function mensajeError_Permisos () {
394     dialog --clear --title 'Permisos' \
395         --backtitle "Proyecto UNIX: Femat Delgado, Mendez Lara, Muñoz Cerda, Prieto Garcia" \
396         --msgbox 'No tiene permisos para realizar la accion' 6 30 --stdout
397     hacerRoot "$1"
398 }
399
400 function mensajeError_Root () {
401     dialog --clear --title 'Permisos' \
402         --backtitle "Proyecto UNIX: Femat Delgado, Mendez Lara, Muñoz Cerda, Prieto Garcia" \
403         --msgbox 'Debe ser root para cambiar el propietario' 6 30 --stdout
404     hacerRoot "$1"
405 }
406
407 function hacerRoot () {
408     dialog --clear --title 'Permisos' \
409         --backtitle "Proyecto UNIX: Femat Delgado, Mendez Lara, Muñoz Cerda, Prieto Garcia" \
410         --yesno 'Quiere volverse usuario root?' 6 30 --stdout
411     Op=$?
412     if [ "$Op" -eq 0 ]; then
413         dialog --clear
414         clear
415         if [ "$(sudo -s ./main.sh \"$1\" \"modo root\")" -ne "0" ]; then
416             dialog --clear --title "Permisos" \
417                 --backtitle "Proyecto UNIX: Femat Delgado, Mendez Lara, Muñoz Cerda, Prieto Garcia" \
418                 --msgbox "No se pudo volver root" 0 0
419             Mostrar_archivos "$1"
```

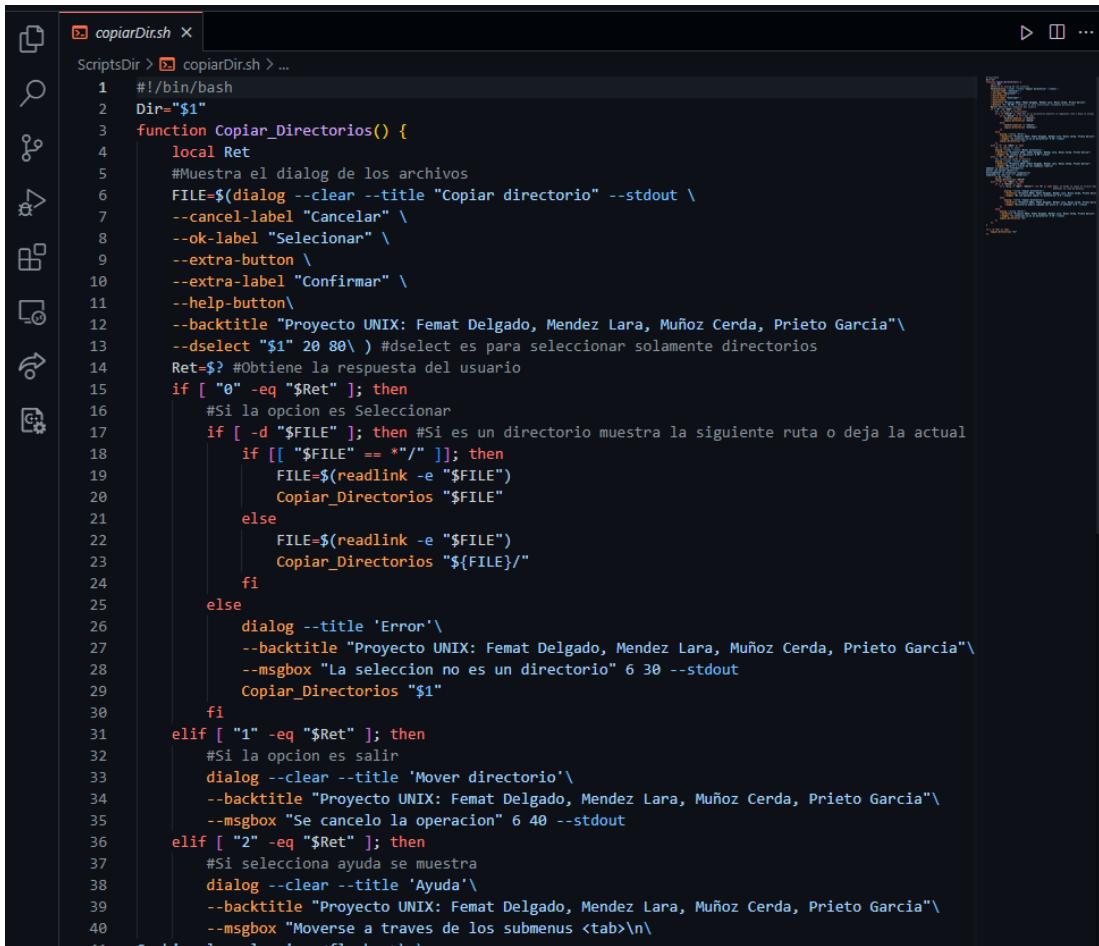
```
main.sh - Códigos - Visual Studio Code
```

```
main.sh
```

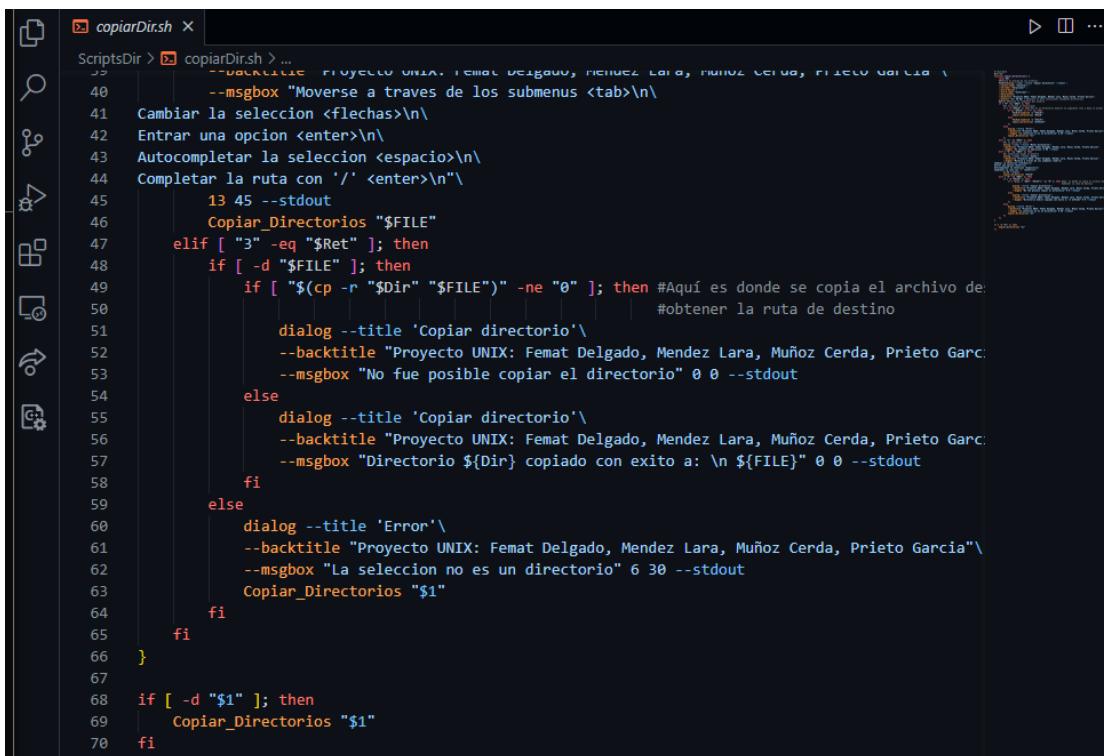
```
428
429  #Main script comienza aqui----- (No necesita modificaciones, a excepcion de la carpeta de trabajo)
430  if [ -z "$1" ]; then
431      #Si la entrada es nula, coloca la ruta predeterminada
432      if [ "$(whoami)" == 'root' ]; then
433          Filepath="/home/"
434      else
435          Filepath="/home/$(whoami)"/ #Selecciona la carpeta del usuario que este usando el comando
436      fi
437      Mostrar_archivos "$Filepath" #Envia el directorio como parametro a mostrar_archivos
438  elif [ -d "$1" ]; then
439      if [ "$2" == 'modo root' ]; then
440          dialog --clear --title "Permisos"
441          --backtitle "Proyecto UNIX: Femat Delgado, Mendez Lara, Muñoz Cerda, Prieto Garcia"
442          --msgbox "Accedio a modo super usuario, cuando termine se le retornara al programa donde se ejecuto"
443          Mostrar_archivos "$1"
444      fi
445      #Si el archivo es un directorio
446      Filepath="$1"
447      SeleccionOpciones "$1" #Manda a llamar a mostrar_archivos con el mismo directorio que se envio
448  elif [ -f "$1" ]; then
449      if [ "$2" == 'modo root' ]; then
450          dialog --clear --title "Permisos"
451          --backtitle "Proyecto UNIX: Femat Delgado, Mendez Lara, Muñoz Cerda, Prieto Garcia"
452          --msgbox "Accedio a modo super usuario, cuando termine se le retornara al programa donde se ejecuto"
453          Mostrar_archivos "$1"
454      fi
455      Filepath="$1"
456      SeleccionOpciones "$1" #Si es un archivo el que se manda, llama a la funcion de mostrar opciones
457  else
458      Filepath="$1"
459      Mostrar_archivos /home/ #Si ninguna opcion es valida, simplemente muestra los archivos en la carpeta de trabajo
460  fi
461  dialog --clear
462  clear
463
464  #Final del script-----
```

No se encuentra Make.

- **copiarDir:** Copia un archivo y obtiene la ruta del destino a donde se quiera copiar.

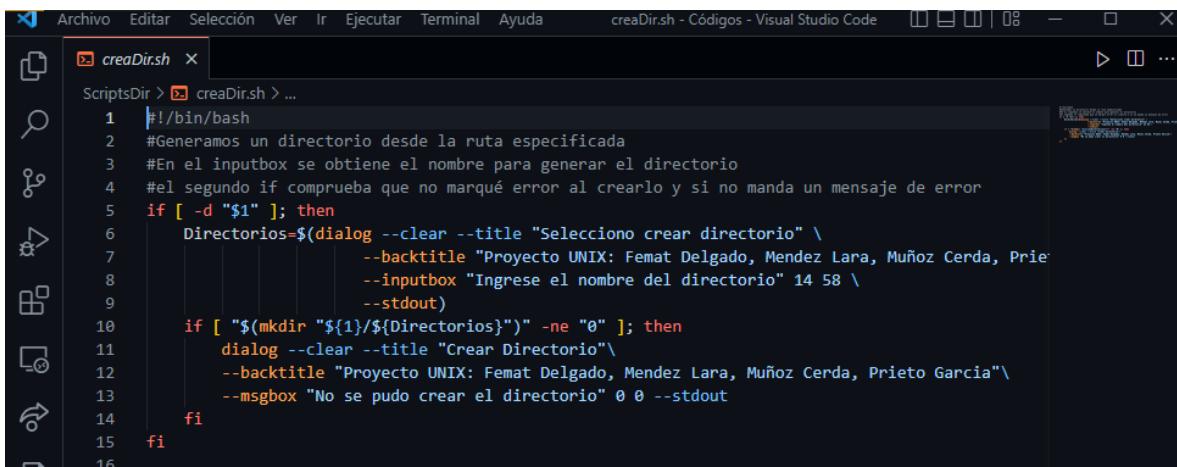


```
copiarDir.sh
ScriptsDir > copiarDir.sh > ...
1  #!/bin/bash
2  Dir="$1"
3  function Copiar_Directorios() {
4      local Ret
5      #Muestra el dialog de los archivos
6      FILE=$(dialog --clear --title "Copiar directorio" --stdout \
7          --cancel-label "Cancelar" \
8          --ok-label "Seleccionar" \
9          --extra-button \
10         --extra-label "Confirmar" \
11         --help-button \
12         --backtitle "Proyecto UNIX: Femat Delgado, Mendez Lara, Muñoz Cerda, Prieto Garcia" \
13         --select "$1" 20 80 ) #select es para seleccionar solamente directorios
14      Ret=$? #Obtiene la respuesta del usuario
15      if [ "0" -eq "$Ret" ]; then
16          #Si la opcion es Seleccionar
17          if [ -d "$FILE" ]; then #Si es un directorio muestra la siguiente ruta o deja la actual
18              if [[ "$FILE" == "/" ]]; then
19                  FILE=$(readlink -e "$FILE")
20                  Copiar_Directorios "$FILE"
21              else
22                  FILE=$(readlink -e "$FILE")
23                  Copiar_Directorios "${FILE}/"
24              fi
25          else
26              dialog --title 'Error' \
27              --backtitle "Proyecto UNIX: Femat Delgado, Mendez Lara, Muñoz Cerda, Prieto Garcia" \
28              --msgbox "La seleccion no es un directorio" 6 30 --stdout
29              Copiar_Directorios "$1"
30          fi
31      elif [ "1" -eq "$Ret" ]; then
32          #Si la opcion es salir
33          dialog --clear --title 'Mover directorio' \
34          --backtitle "Proyecto UNIX: Femat Delgado, Mendez Lara, Muñoz Cerda, Prieto Garcia" \
35          --msgbox "Se cancelo la operacion" 6 40 --stdout
36      elif [ "2" -eq "$Ret" ]; then
37          #Si selecciona ayuda se muestra
38          dialog --clear --title 'Ayuda' \
39          --backtitle "Proyecto UNIX: Femat Delgado, Mendez Lara, Muñoz Cerda, Prieto Garcia" \
40          --msgbox "Moverse a traves de los submenu's <tab>\n"
41      fi
42  }
43  Copiar_Directorios "$1"
```



```
copiarDir.sh
ScriptsDir > copiarDir.sh > ...
40      --msgbox "Moverse a traves de los submenu's <tab>\n"
41  Cambiar la seleccion <flechas>\n
42  Entrar una opcion <enter>\n
43  Autocompletar la seleccion <espacio>\n
44  Completar la ruta con '/' <enter>\n"
45      13 45 --stdout
46      Copiar_Directorios "$FILE"
47  elif [ "3" -eq "$Ret" ]; then
48      if [ -d "$FILE" ]; then
49          if [ "$(cp -r "$Dir" "$FILE")" -ne "0" ]; then #Aqui es donde se copia el archivo de:
50              #obtener la ruta de destino
51              dialog --title 'Copiar directorio' \
52              --backtitle "Proyecto UNIX: Femat Delgado, Mendez Lara, Muñoz Cerda, Prieto Garcia" \
53              --msgbox "No fue posible copiar el directorio" 0 0 --stdout
54      else
55          dialog --title 'Copiar directorio' \
56          --backtitle "Proyecto UNIX: Femat Delgado, Mendez Lara, Muñoz Cerda, Prieto Garcia" \
57          --msgbox "Directorio ${Dir} copiado con exito a: \n ${FILE}" 0 0 --stdout
58      fi
59  else
60      dialog --title 'Error' \
61      --backtitle "Proyecto UNIX: Femat Delgado, Mendez Lara, Muñoz Cerda, Prieto Garcia" \
62      --msgbox "La seleccion no es un directorio" 6 30 --stdout
63      Copiar_Directorios "$1"
64  fi
65  fi
66  }
67
68  if [ -d "$1" ]; then
69      Copiar_Directorios "$1"
70  fi
```

- **creaDir:** Se crea un directorio desde la ruta donde se especifica.

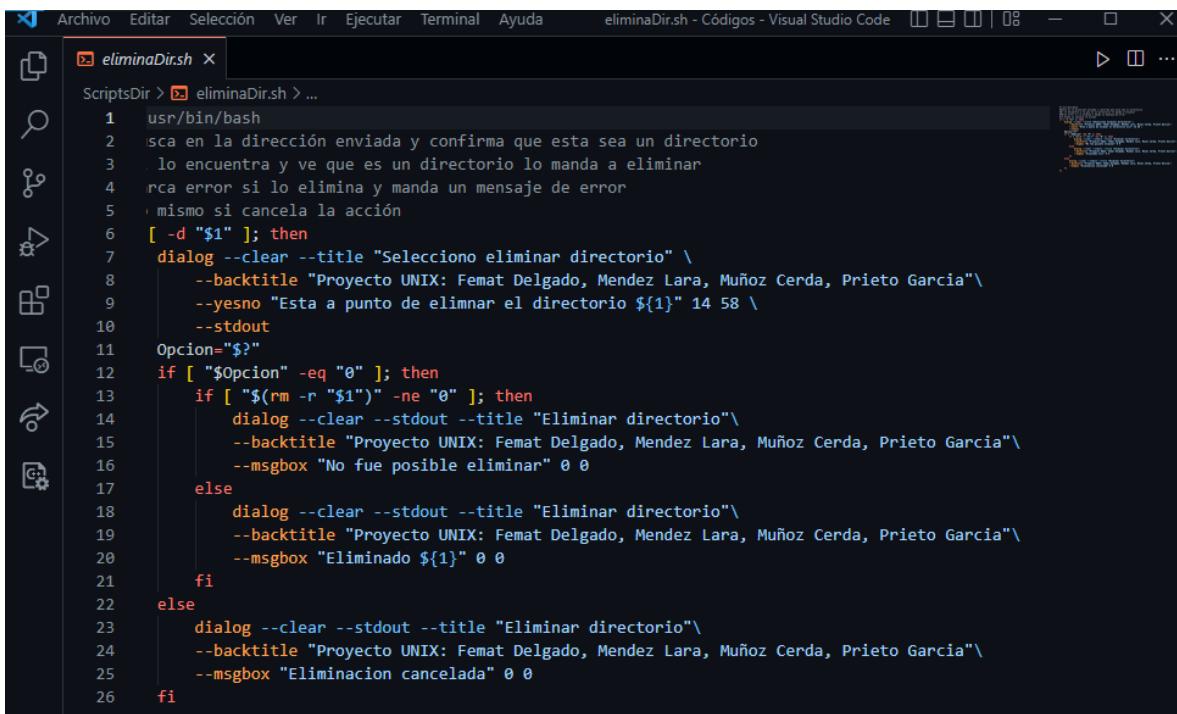


```

1 #!/bin/bash
2 #Generamos un directorio desde la ruta especificada
3 #En el inputbox se obtiene el nombre para generar el directorio
4 #el segundo if comprueba que no marqué error al crearlo y si no manda un mensaje de error
5 if [ -d "$1" ]; then
6     Directorios=$(dialog --clear --title "Selecciono crear directorio" \
7                         --backtitle "Proyecto UNIX: Femat Delgado, Mendez Lara, Muñoz Cerda, Prieto Garcia" \
8                         --inputbox "Ingrese el nombre del directorio" 14 58 \
9                         --stdout)
10    if [ "$(mkdir "${1}/${Directorios}")" -ne "0" ]; then
11        dialog --clear --title "Crear Directorio" \
12                         --backtitle "Proyecto UNIX: Femat Delgado, Mendez Lara, Muñoz Cerda, Prieto Garcia" \
13                         --msgbox "No se pudo crear el directorio" 0 0 --stdout
14    fi
15 fi
16

```

- **eliminaDir:** Verifica la dirección que se le envió y verifica que este sea un directorio, después de encontrarlo y confirmarlo lo elimina.



```

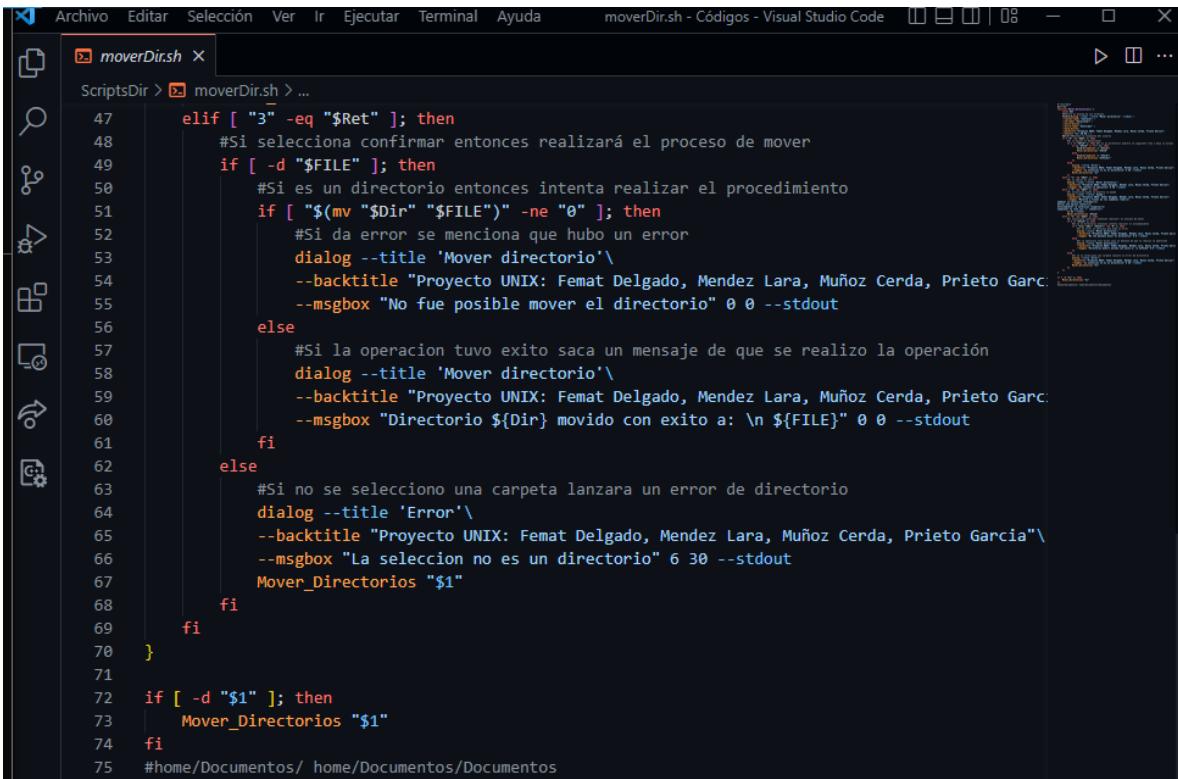
1 usr/bin/bash
2 saca en la dirección enviada y confirma que esta sea un directorio
3 lo encuentra y ve que es un directorio lo manda a eliminar
4 manda error si lo elimina y manda un mensaje de error
5 mismo si cancela la acción
6 [ -d "$1" ]; then
7     dialog --clear --title "Selecciono eliminar directorio" \
8                         --backtitle "Proyecto UNIX: Femat Delgado, Mendez Lara, Muñoz Cerda, Prieto Garcia" \
9                         --yesno "Esta a punto de eliminar el directorio ${1}" 14 58 \
10                        --stdout
11 Opcion="$?"
12 if [ "$Opcion" -eq "0" ]; then
13     if [ "$(rm -r "$1")" -ne "0" ]; then
14         dialog --clear --stdout --title "Eliminar directorio" \
15                         --backtitle "Proyecto UNIX: Femat Delgado, Mendez Lara, Muñoz Cerda, Prieto Garcia" \
16                         --msgbox "No fue posible eliminar" 0 0
17     else
18         dialog --clear --stdout --title "Eliminar directorio" \
19                         --backtitle "Proyecto UNIX: Femat Delgado, Mendez Lara, Muñoz Cerda, Prieto Garcia" \
20                         --msgbox "Eliminado ${1}" 0 0
21     fi
22 else
23     dialog --clear --stdout --title "Eliminar directorio" \
24                         --backtitle "Proyecto UNIX: Femat Delgado, Mendez Lara, Muñoz Cerda, Prieto Garcia" \
25                         --msgbox "Eliminacion cancelada" 0 0
26 fi
27

```

- **moverDir:** Mueve un directorio con todo su contenido a una nueva dirección.

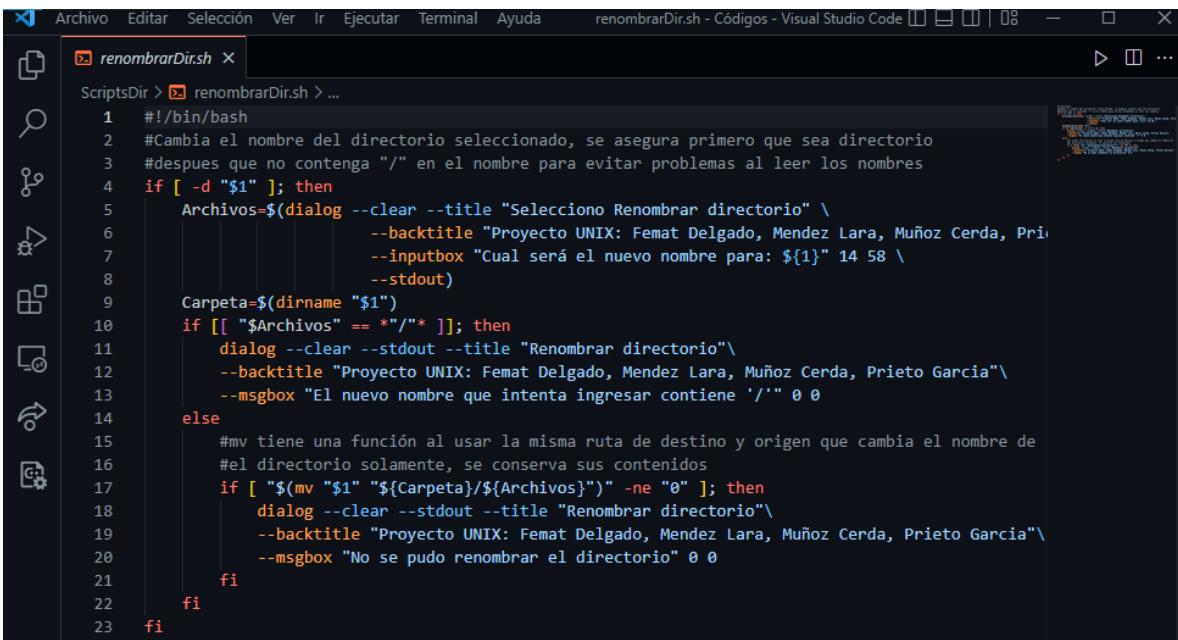
The screenshot shows a terminal window within Visual Studio Code, which is itself running on a desktop environment with a gold-colored decorative border featuring circular patterns. The terminal window has a dark theme and displays a Bash script named `moverDir.sh`. The script contains several conditional statements (if, elif, else) and functions, primarily using the `dialog` command to interact with the user via a graphical interface. The code is well-formatted with line numbers on the left.

```
#!/bin/bash
Dir="$1"
function Mover_Directorios() {
    local Ret
    #Muestra el dialog de los archivos
    FILE=$(dialog --clear --title "Mover directorio" --stdout \
    --cancel-label "Cancelar" \
    --ok-label "Seleccionar" \
    --extra-button \
    --extra-label "Confirmar" \
    --help-button \
    --backtitle "Proyecto UNIX: Femat Delgado, Mendez Lara, Muñoz Cerda, Prieto Garcia" \
    --select "$1" 20 80 \
)
Ret=$? #Obtiene la respuesta del usuario
if [ "0" -eq "$Ret" ]; then
    #Si la opcion es Seleccionar
    if [ -d "$FILE" ]; then #Si es un directorio muestra la siguiente ruta o deja la actual
        if [[ "$FILE" == */ ]]; then
            FILE=$(readlink -e "$FILE")
            Mover_Directorios "$FILE"
        else
            FILE=$(readlink -e "$FILE")
            Mover_Directorios "${FILE}/"
        fi
    else
        dialog --title 'Error' \
        --backtitle "Proyecto UNIX: Femat Delgado, Mendez Lara, Muñoz Cerda, Prieto Garcia" \
        --msgbox "La seleccion no es un directorio" 6 30 --stdout
        Mover_Directorios "$1"
    fi
elif [ "1" -eq "$Ret" ]; then
    #Si la opcion es salir
    dialog --clear --title 'Mover directorio' \
    --backtitle "Proyecto UNIX: Femat Delgado, Mendez Lara, Muñoz Cerda, Prieto Garcia" \
    --msgbox "Se cancelo la operacion" 6 40 --stdout
elif [ "2" -eq "$Ret" ]; then
    #Si selecciona ayuda se muestra la ayuda
    dialog --clear --title 'Ayuda' \
    --backtitle "Proyecto UNIX: Femat Delgado, Mendez Lara, Muñoz Cerda, Prieto Garcia" \
    --msgbox "Moverse a traves de los submenus <tab>\n"
Cambiar la seleccion <flechas>\n
Entrar una opcion <enter>\n
Autocompletar la seleccion <espacio>\n
Completar la ruta con '/' <enter>\n\
    13 45 --stdout
    Mover_Directorios "$FILE"
elif [ "3" -eq "$Ret" ]; then
    #Si selecciona confirmar entonces realizará el proceso de mover
    if [ -d "$FILE" ]; then
        #Si es un directorio entonces intenta realizar el procedimiento
        if [ -d "$FILE" ]; then
            dialog --clear --title 'Confirmar' \
            --backtitle "Proyecto UNIX: Femat Delgado, Mendez Lara, Muñoz Cerda, Prieto Garcia" \
            --msgbox "¿Desea mover el directorio '$FILE' a la ruta seleccionada?" 6 30 --stdout
            Ret=$?
            if [ "0" -eq "$Ret" ]; then
                dialog --clear --title 'Confirmacion' \
                --backtitle "Proyecto UNIX: Femat Delgado, Mendez Lara, Muñoz Cerda, Prieto Garcia" \
                --msgbox "El directorio '$FILE' ha sido movido exitosamente" 6 30 --stdout
            else
                dialog --clear --title 'Error' \
                --backtitle "Proyecto UNIX: Femat Delgado, Mendez Lara, Muñoz Cerda, Prieto Garcia" \
                --msgbox "No se pudo mover el directorio '$FILE'" 6 30 --stdout
            fi
        else
            dialog --clear --title 'Error' \
            --backtitle "Proyecto UNIX: Femat Delgado, Mendez Lara, Muñoz Cerda, Prieto Garcia" \
            --msgbox "No se pudo mover el directorio '$FILE' porque no es un directorio" 6 30 --stdout
        fi
    else
        dialog --clear --title 'Error' \
        --backtitle "Proyecto UNIX: Femat Delgado, Mendez Lara, Muñoz Cerda, Prieto Garcia" \
        --msgbox "No se pudo mover el directorio '$FILE' porque no es un directorio" 6 30 --stdout
    fi
fi
}
Mover_Directorios "$1"
```



```
#!/bin/bash
#Si selecciona confirmar entonces realizará el proceso de mover
if [ -d "$FILE" ]; then
    #Si es un directorio entonces intenta realizar el procedimiento
    if [ $(mv "$Dir" "$FILE") -ne "0" ]; then
        #Si da error se menciona que hubo un error
        dialog --title 'Mover directorio' \
        --backtitle "Proyecto UNIX: Femat Delgado, Mendez Lara, Muñoz Cerda, Prieto Garcia" \
        --msgbox "No fue posible mover el directorio" 0 0 --stdout
    else
        #Si la operacion tuvo exito saca un mensaje de que se realizo la operación
        dialog --title 'Mover directorio' \
        --backtitle "Proyecto UNIX: Femat Delgado, Mendez Lara, Muñoz Cerda, Prieto Garcia" \
        --msgbox "Directorio ${Dir} movido con exito a: \n ${FILE}" 0 0 --stdout
    fi
else
    #Si no se selecciono una carpeta lanzara un error de directorio
    dialog --title 'Error' \
    --backtitle "Proyecto UNIX: Femat Delgado, Mendez Lara, Muñoz Cerda, Prieto Garcia" \
    --msgbox "La seleccion no es un directorio" 6 30 --stdout
    Mover_Directorios "$1"
fi
fi
}
if [ -d "$1" ]; then
    Mover_Directorios "$1"
fi
#home/Documentos/ home/Documentos/Documentos
```

- **renombrarDir:** Cambia el nombre de un directorio.

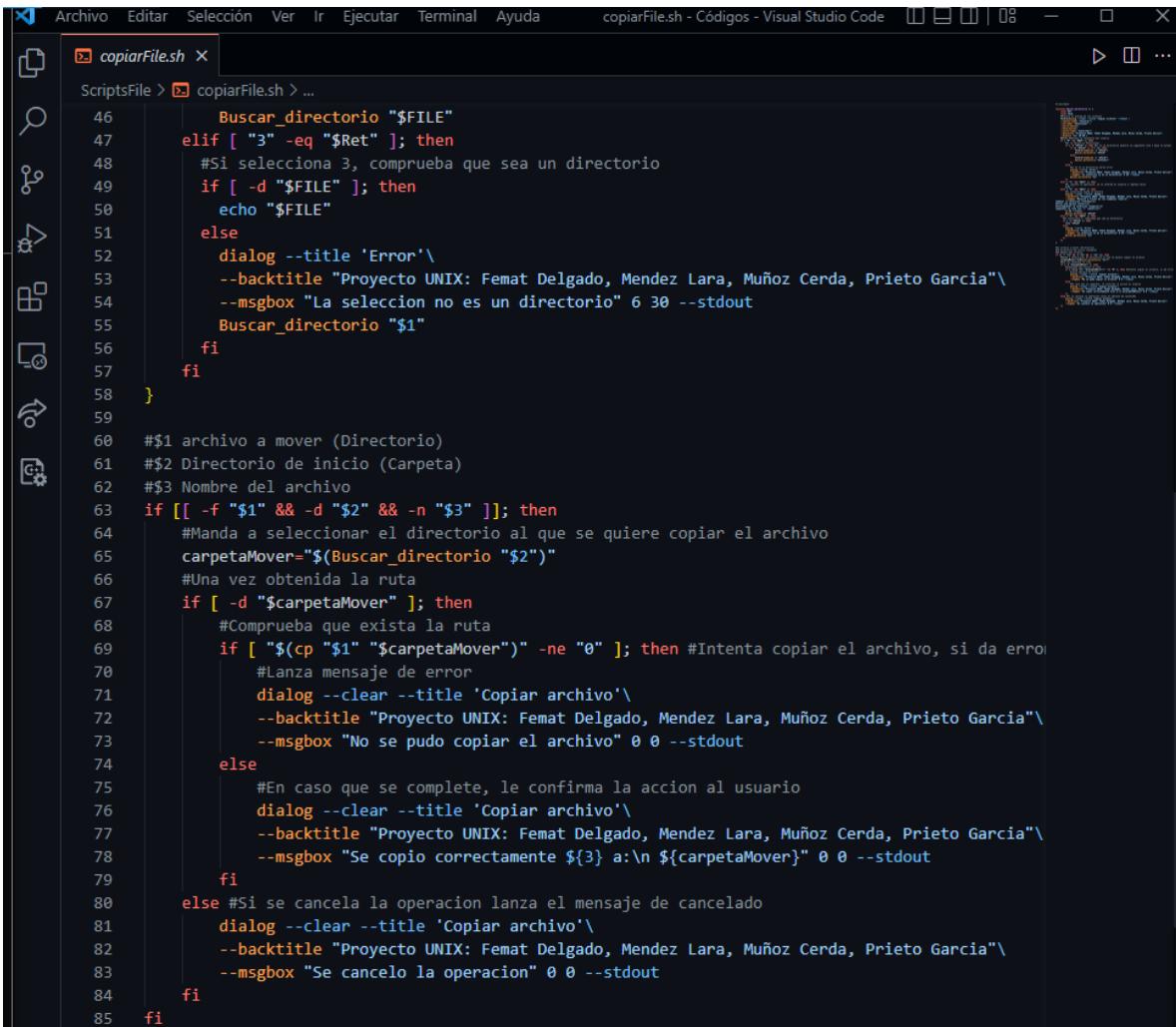


```
#!/bin/bash
#Cambia el nombre del directorio seleccionado, se asegura primero que sea directorio
#despues que no contenga "/" en el nombre para evitar problemas al leer los nombres
if [ -d "$1" ]; then
    Archivos=$(dialog --clear --title "Selecciono Renombrar directorio" \
    --backtitle "Proyecto UNIX: Femat Delgado, Mendez Lara, Muñoz Cerda, Prieto Garcia" \
    --inputbox "Cual será el nuevo nombre para: ${1}" 14 58 \
    --stdout)
    Carpeta=$(dirname "$1")
    if [[ "$Archivos" == */* ]]; then
        dialog --clear --stdout --title "Renombrar directorio" \
        --backtitle "Proyecto UNIX: Femat Delgado, Mendez Lara, Muñoz Cerda, Prieto Garcia" \
        --msgbox "El nuevo nombre que intenta ingresar contiene '/'" 0 0
    else
        #mv tiene una función al usar la misma ruta de destino y origen que cambia el nombre de
        #el directorio solamente, se conserva sus contenidos
        if [ $(mv "$1" "${Carpeta}/${Archivos}") -ne "0" ]; then
            dialog --clear --stdout --title "Renombrar directorio" \
            --backtitle "Proyecto UNIX: Femat Delgado, Mendez Lara, Muñoz Cerda, Prieto Garcia" \
            --msgbox "No se pudo renombrar el directorio" 0 0
        fi
    fi
fi
```

- **copiarFile:** Copia un archivo y obtiene la ruta del destino a donde se quiera copiar.

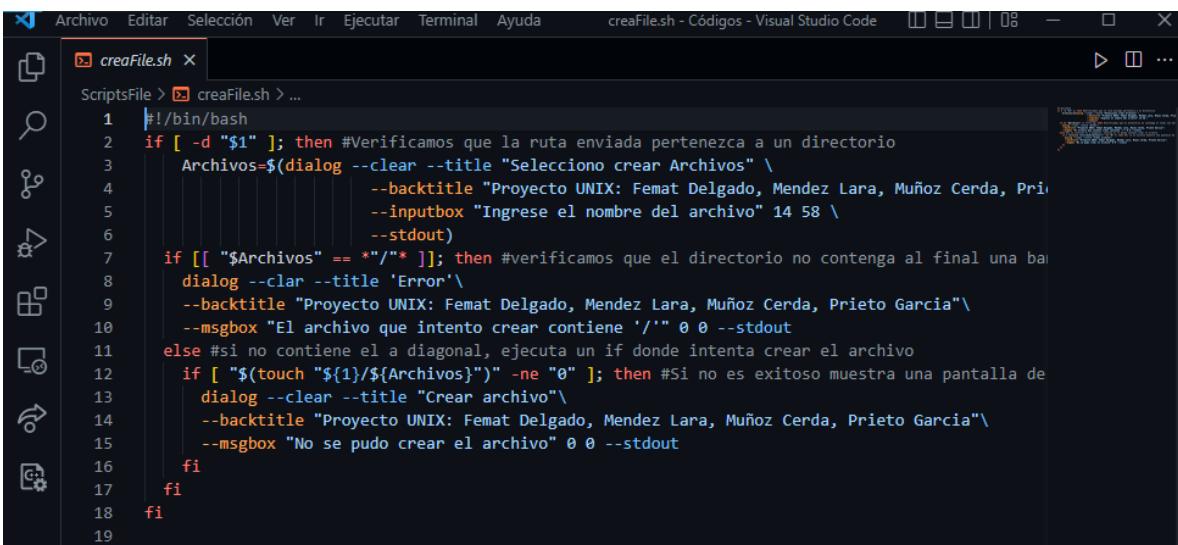
```
#!/bin/bash

function Buscar_directorio () {
    local Ret
    local FILE
    #Muestra el dialog de los archivos
    FILE=$(dialog --clear --title "Copiar archivo" --stdout \
    --cancel-label "Cancelar" \
    --ok-label "Seleccionar" \
    --extra-button \
    --help-button \
    --extra-label "Confirmar" \
    --backtitle "Proyecto UNIX: Femat Delgado, Mendez Lara, Muñoz Cerda, Prieto Garcia" \
    --select "$1" 20 80 )
    Ret=$? #Obtiene la respuesta del usuario
    if [ "$0" -eq "$Ret" ]; then
        #Si la opcion es Seleccionar
        if [ -d "$FILE" ]; then #Si es un directorio muestra la siguiente ruta o deja la actual
            if [[ "$FILE" == *"/" ]]; then
                FILE=$(readlink -e "$FILE")
                Buscar_directorio "$FILE"
            else
                FILE=$(readlink -e "$FILE")
                Buscar_directorio "${FILE}/"
            fi
        else
            #Si no es un directorio marca error
            dialog --title 'Error' \
            --backtitle "Proyecto UNIX: Femat Delgado, Mendez Lara, Muñoz Cerda, Prieto Garcia" \
            --msgbox "La seleccion no es un directorio" 6 30 --stdout
            Buscar_directorio "$1"
        fi
    elif [ "1" -eq "$Ret" ]; then
        #Si cancelo la operacion, se lo informa al usuario y regresa vacio
        echo ''
    elif [ "2" -eq "$Ret" ]; then
        #Si selecciona ayuda se muestra
        dialog --clear --title 'Ayuda' \
        --backtitle "Proyecto UNIX: Femat Delgado, Mendez Lara, Muñoz Cerda, Prieto Garcia" \
        --msgbox "Moverse a traves de los submenus <tab>\n\nCambiar la seleccion <flechas>\n\nEntrar una opcion <enter>\n\nAutocompletar la seleccion <espacio>\n\nCompletar la ruta con '/' <enter>\n\n13 45 --stdout
        Buscar_directorio "$FILE"
    elif [ "3" -eq "$Ret" ]; then
        #Si selecciona 3, comprueba que sea un directorio
        if [ -d "$FILE" ]; then
            echo "$FILE"
        fi
    fi
}
```



```
46     Buscar_directorio "$FILE"
47     elif [ "$3" -eq "$Ret" ]; then
48         #Si selecciona 3, comprueba que sea un directorio
49         if [ -d "$FILE" ]; then
50             echo "$FILE"
51         else
52             dialog --title 'Error'\n                --backtitle "Proyecto UNIX: Femat Delgado, Mendez Lara, Muñoz Cerdá, Prieto García"\n                --msgbox "La selección no es un directorio" 6 30 --stdout
53             Buscar_directorio "$1"
54         fi
55     fi
56 }
57
58 }
59
60 ##$1 archivo a mover (Directorio)
61 ##$2 Directorio de inicio (Carpeta)
62 ##$3 Nombre del archivo
63 if [[ -f "$1" && -d "$2" && -n "$3" ]]; then
64     #Manda a seleccionar el directorio al que se quiere copiar el archivo
65     carpetaMover=$(Buscar_directorio "$2")
66     #Una vez obtenida la ruta
67     if [ -d "$carpetaMover" ]; then
68         #Comprueba que exista la ruta
69         if [ "$(cp "$1" "$carpetaMover")" -ne "0" ]; then #Intenta copiar el archivo, si da error
70             #Lanza mensaje de error
71             dialog --clear --title 'Copiar archivo'
72             --backtitle "Proyecto UNIX: Femat Delgado, Mendez Lara, Muñoz Cerdá, Prieto García"\n                --msgbox "No se pudo copiar el archivo" 0 0 --stdout
73         else
74             #En caso que se complete, le confirma la acción al usuario
75             dialog --clear --title 'Copiar archivo'\n                --backtitle "Proyecto UNIX: Femat Delgado, Mendez Lara, Muñoz Cerdá, Prieto García"\n                --msgbox "Se copió correctamente ${3} a:\n${carpetaMover}" 0 0 --stdout
76         fi
77     else #Si se cancela la operación lanza el mensaje de cancelado
78         dialog --clear --title 'Copiar archivo'\n            --backtitle "Proyecto UNIX: Femat Delgado, Mendez Lara, Muñoz Cerdá, Prieto García"\n            --msgbox "Se canceló la operación" 0 0 --stdout
79     fi
80 fi
81 fi
82
83 fi
84 fi
85 fi
```

- **creaFile:** Primero verifica que la ruta enviada pertenezca a un directorio, de comprobarse crea un archivo. En caso de que falle la acción, mostrara un mensaje con error.



```
1 #!/bin/bash
2 if [ -d "$1" ]; then #Verificamos que la ruta enviada pertenezca a un directorio
3     Archivos=$(dialog --clear --title "Selecciona crear Archivos" \
4                         --backtitle "Proyecto UNIX: Femat Delgado, Mendez Lara, Muñoz Cerdá, Prieto García"\n                --inputbox "Ingrese el nombre del archivo" 14 58 \
5                         --stdout)
6
7 if [[ "$Archivos" == /*/* ]]; then #verificamos que el directorio no contenga al final una barra diagonal
8     dialog --clear --title 'Error'\n        --backtitle "Proyecto UNIX: Femat Delgado, Mendez Lara, Muñoz Cerdá, Prieto García"\n        --msgbox "El archivo que intento crear contiene '/'" 0 0 --stdout
9
10 else #si no contiene el a diagonal, ejecuta un if donde intenta crear el archivo
11     if [ "$(touch "${1}/${Archivos}")" -ne "0" ]; then #Si no es exitoso muestra una pantalla de error
12         dialog --clear --title "Crear archivo"\n            --backtitle "Proyecto UNIX: Femat Delgado, Mendez Lara, Muñoz Cerdá, Prieto García"\n            --msgbox "No se pudo crear el archivo" 0 0 --stdout
13     fi
14
15 fi
16
17 fi
18 fi
19
```

- **editarArchivos:** Modifica el contenido del archivo al que se le vaya a editar, por el contenido que se le mande.



```

1  #!/bin/bash
2
3  if [[ -f "$1" && -d "$2" && -n "$3" ]]; then #Checa si hay parametros
4      Escoger=$(dialog --clear --title 'Editar Archivo' \
5          --backtitle "Proyecto UNIX: Femat Delgado, Mendez Lara, Muñoz Cerda, Prieto Garcia" \
6          --radiolist "Seleccione un editor de texto para : ${1}" 0 0 0 \
7          1 'Integrado' 1 \
8          2 'Nano' 2 \
9          3 'Vi' 3 \
10         --stdout)
11  if [ "${?}" -eq 0 ]; then
12      if [ "$Escoger" -eq 1 ]; then
13          CONTENIDOARCHIVO=$(dialog --clear --title "Editando ${3}" \
14              --backtitle "Proyecto UNIX: Femat Delgado, Mendez Lara, Muñoz Cerda, Prieto Garcia" \
15              --editbox "$1" --stdout 40 100)" #Se crea dialog con un editbox que altera el contenido del archivo
16          Opcion=${?}
17          if [[ "${Opcion}" == "0" ]]; then
18              touch "${2}/tmp_${3}" #Se crea archivo temporal con el nombre del archivo
19              echo "$CONTENIDOARCHIVO">"${2}/tmp_${3}" #Se envia el contenido del editbox al archivo temporal
20              mv "${2}/tmp_${3}" "$1" #Se le cambia el nombre al archivo temporal por el nombre del archivo original
21              rm "${2}/tmp_${3}"
22          else
23              dialog --clear --title "Edicion ${3}" \
24                  --backtitle "Proyecto UNIX: Femat Delgado, Mendez Lara, Muñoz Cerda, Prieto Garcia" \
25                  --msgbox "Se cancelo la edicion del archivo" 0 0 --stdout
26          fi
27      elif [ "$Escoger" -eq 2 ]; then
28          nano "$1"
29      elif [ "$Escoger" -eq 3 ]; then
30          vi "$1"
31      fi
32  else
33      dialog --clear --title "Edicion" \
34          --backtitle "Proyecto UNIX: Femat Delgado, Mendez Lara, Muñoz Cerda, Prieto Garcia" \
35          --msgbox "Edicion cancelada" 0 0 --stdout
36  fi
37 fi

```

- **eliminaFile:** Primero comprueba que desde la ruta donde se solicita la función sea un archivo, para después de confirmarlo manda un mensaje confirmando que se desea eliminar ese archivo, en caso de que la respuesta sea afirmativa se elimina, de no serlo se cancela la acción.



```

1  ScriptsFile > eliminaFile.sh > ...
2  #!/usr/bin/bash
3  if [ -f "$1" ]; then #Verificamos que la ruta mandada sea un archivo, en caso de serlo manda un mensaje de confirmación si quiere eliminarlo
4      dialog --clear --title "Selecciono eliminar archivo" \
5          --backtitle "Proyecto UNIX: Femat Delgado, Mendez Lara, Muñoz Cerda, Prieto Garcia" \
6          --yesno "Esta a punto de eliminar el archivo ${1}" 14 58 \
7          --stdout
8      Opcion=${?}
9      if [ "$Opcion" -eq 0 ]; then # si la respuesta es positiva, regresa un 0 y elimina el archivo
10         if [ "$(rm "$1")" -ne "0" ]; then #Corremos el comando y si no es exitoso mandamos un mensaje de error
11             dialog --clear --title "Eliminar Archivo" \
12                 --backtitle "Proyecto UNIX: Femat Delgado, Mendez Lara, Muñoz Cerda, Prieto Garcia" \
13                 --msgbox "No se pudo eliminar el archivo" 0 0 --stdout
14         fi
15     else #si la respuesta es cancelar, se regresa un 1 y mandamos un mensaje de cancelación
16         dialog --clear --title "Eliminar Archivo" \
17             --backtitle "Proyecto UNIX: Femat Delgado, Mendez Lara, Muñoz Cerda, Prieto Garcia" \
18             --msgbox "Eliminacion cancelada" 0 0 --stdout
19     fi
20 fi

```

- **Mostrar:** Muestra el contenido de un archivo mediante un msgbox.



```
mostrar.sh > ScriptsFile > mostrar.sh > ...
1  #!/bin/bash
2
3  #Muestra el contenido de un archivo mediante un msgbox
4
5  #\$1 Es el archivo que se va a mostrar
6  #\$2 Es el nombre del archivo que se muestra
7
8  if [[ -f "\$1" && -n "\$2" ]]; then
9      Info=$(cat -n "\$1")    #Le hace cat al archivo y lo guarda en una variable
10     if [[ "\${Info}" == * ]]; then
11         dialog --clear --stdout --title "\$2" --ok-label "Continuar" \
12             --backtitle "Proyecto UNIX: Femat Delgado, Mendez Lara, Muñoz Cerda, Prieto Garcia" \
13             --msgbox "\${Info}" 32 100
14         #Saca la informacion del archivo en un msgbox
15     else
16         dialog --clear --stdout --title "\$2" \
17             --backtitle "Proyecto UNIX: Femat Delgado, Mendez Lara, Muñoz Cerda, Prieto Garcia" \
18             --msgbox "No se fue posible leer el archivo" 32 100
19     fi
20 fi
```

- **moveFile:** Mueve un archivo con todo su contenido a una nueva dirección.

```

moverFile.sh
#!/bin/bash

function Buscar_directorio () {
    local Ret
    local FILE
    #Muestra el dialog de los archivos
    FILE=$(dialog --clear --title "Mover archivo" --stdout \
    --cancel-label "Cancelar" \
    --ok-label "Seleccionar" \
    --extra-button \
    --help-button \
    --extra-label "Confirmar" \
    --backtitle "Proyecto UNIX: Femat Delgado, Mendez Lara, Muñoz Cerdá, Prieto García" \
    --dselect "$1" 20 80 )
    Ret=$? #Obtiene la respuesta del usuario
    if [ "0" -eq "$Ret" ]; then
        #Si la opción es Seleccionar
        if [ -d "$FILE" ]; then #Si es un directorio muestra la siguiente ruta o deja la actual
            if [ [ "$FILE" == "/" ]]; then
                FILE=$(readlink -e "$FILE")
                Buscar_directorio "$FILE"
            else
                FILE=$(readlink -e "$FILE")
                Buscar_directorio "${FILE}/"
            fi
        else
            #Si no es un directorio marca error
            dialog --title 'Error' \
            --backtitle "Proyecto UNIX: Femat Delgado, Mendez Lara, Muñoz Cerdá, Prieto García" \
            --msgbox "La selección no es un directorio" 6 30 --stdout
            Buscar_directorio "$1"
        fi
    elif [ "1" -eq "$Ret" ]; then
        #Si canceló la operación, se lo informa al usuario y regresa vacío
        echo ""
    elif [ "2" -eq "$Ret" ]; then
        #Si selecciona ayuda se muestra
        dialog --clear --title 'Ayuda' \
        --backtitle "Proyecto UNIX: Femat Delgado, Mendez Lara, Muñoz Cerdá, Prieto García" \
        --msgbox "Móvete a través de los submenús <tab>\n\nCambiar la selección <flechas>\n\nEntrar una opción <enter>\n\nAutocompletar la selección <espacio>\n\nCompletar la ruta con '/' <enter>\n\n        13 45 --stdout
        Buscar_directorio "$FILE"
    elif [ "3" -eq "$Ret" ]; then
        #Si selecciona 3, comprueba que sea un directorio
        if [ -d "$FILE" ]; then
            echo "$FILE"
        elif [ "3" -eq "$Ret" ]; then
            #Si selecciona 3, comprueba que sea un directorio
            if [ -d "$FILE" ]; then
                echo "$FILE"
            elif [ "3" -eq "$Ret" ]; then
                #Si selecciona 3, comprueba que sea un directorio
                if [ -d "$FILE" ]; then
                    echo "$FILE"
                else
                    dialog --title 'Error' \
                    --backtitle "Proyecto UNIX: Femat Delgado, Mendez Lara, Muñoz Cerdá, Prieto García" \
                    --msgbox "La selección no es un directorio" 6 30 --stdout
                    Buscar_directorio "$1"
                fi
            fi
        fi
    fi
}

#$1 archivo a mover (Directorio)
#$2 Directorio de inicio (Carpeta)
#$3 Nombre del archivo
if [ [ -f "$1" && -d "$2" && -n "$3" ]]; then
    carpetaMover=$(Buscar_directorio "$2")
    if [ -n "$carpetaMover" ]; then
        if [ $(mv "$1" "$carpetaMover") -ne "0" ]; then
            dialog --clear --stdout --title "Mover archivo" \
            --backtitle "Proyecto UNIX: Femat Delgado, Mendez Lara, Muñoz Cerdá, Prieto García" \
            --msgbox "No se pudo mover el archivo" 0 0
        else
            dialog --clear --stdout --title "Mover archivo" \
            --backtitle "Proyecto UNIX: Femat Delgado, Mendez Lara, Muñoz Cerdá, Prieto García" \
            --msgbox "Se movió correctamente ${3} a:\n ${carpetaMover}" 0 0
        fi
    else
        dialog --clear --title "Mover archivo" \
        --backtitle "Proyecto UNIX: Femat Delgado, Mendez Lara, Muñoz Cerdá, Prieto García" \
        --msgbox "Se canceló la operación" 6 40 --stdout
    fi
fi

```

- **renombrarFile:** Cambia el nombre de un archivo.

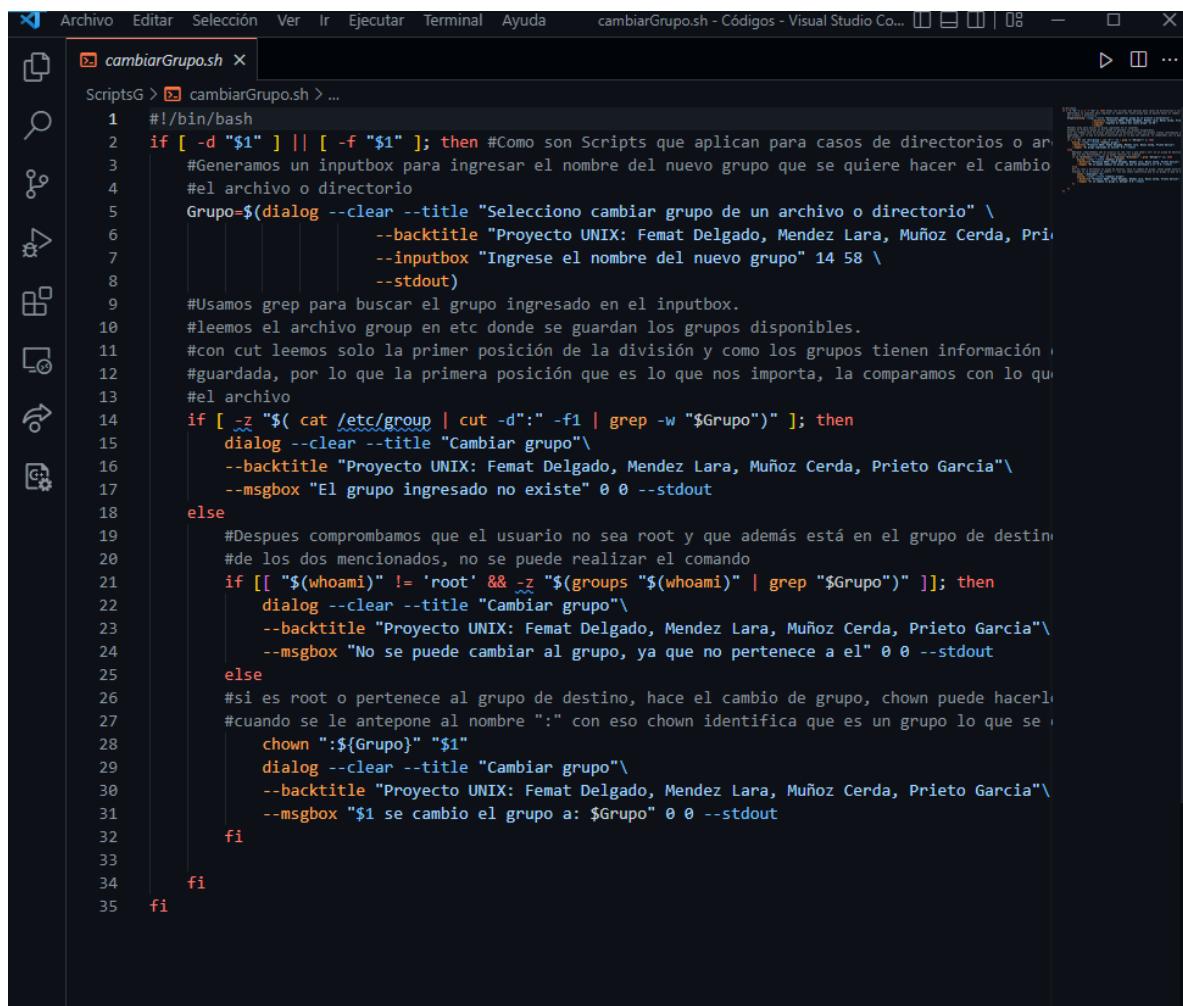


```

renombrarFile.sh
ScriptsFile > renombrarFile.sh > ...
1  #!/bin/bash
2  if [ -f "$1" ]; then #Si la ruta es un archivo, se ejecuta un inputbox el cual tendrá el nombre
3      Archivos=$(dialog --clear --title "Selecciono Renombrar archivo" \
4          --backtitle "Proyecto UNIX: Femat Delgado, Mendez Lara, Muñoz Cerda, Prieto Garcia" \
5          --inputbox "Cual será el nuevo nombre para: ${1}" 14 58 \
6          --stdout)
7  Carpeta=$(dirname "$1") #Generamos una variable con dirname para solo tener la ruta del archivo
8  if [[ "$Archivos" == */* ]]; then #Verificamos que el archivo no contenga una barra diagonal
9      dialog --clear --stdout --title "Renombrar archivo" \
10         --backtitle "Proyecto UNIX: Femat Delgado, Mendez Lara, Muñoz Cerda, Prieto Garcia" \
11         --msgbox "El nuevo nombre que intenta ingresar contiene '/' 0 0
12 else
13     if [ $(mv "$1" "${Carpeta}/${Archivos}") -ne 0 ]; then #ejecutamos el comando pero si falla
14         dialog --clear --stdout --title "Renombrar archivo" \
15             --backtitle "Proyecto UNIX: Femat Delgado, Mendez Lara, Muñoz Cerda, Prieto Garcia" \
16             --msgbox "No se pudo renombrar el archivo" 0 0
17     fi
18 fi
19 fi

```

- **cambiarGrupo:** Si el usuario que solicita la acción es root o pertenece al grupo de destino, hace el cambio de grupo con el comando chown.

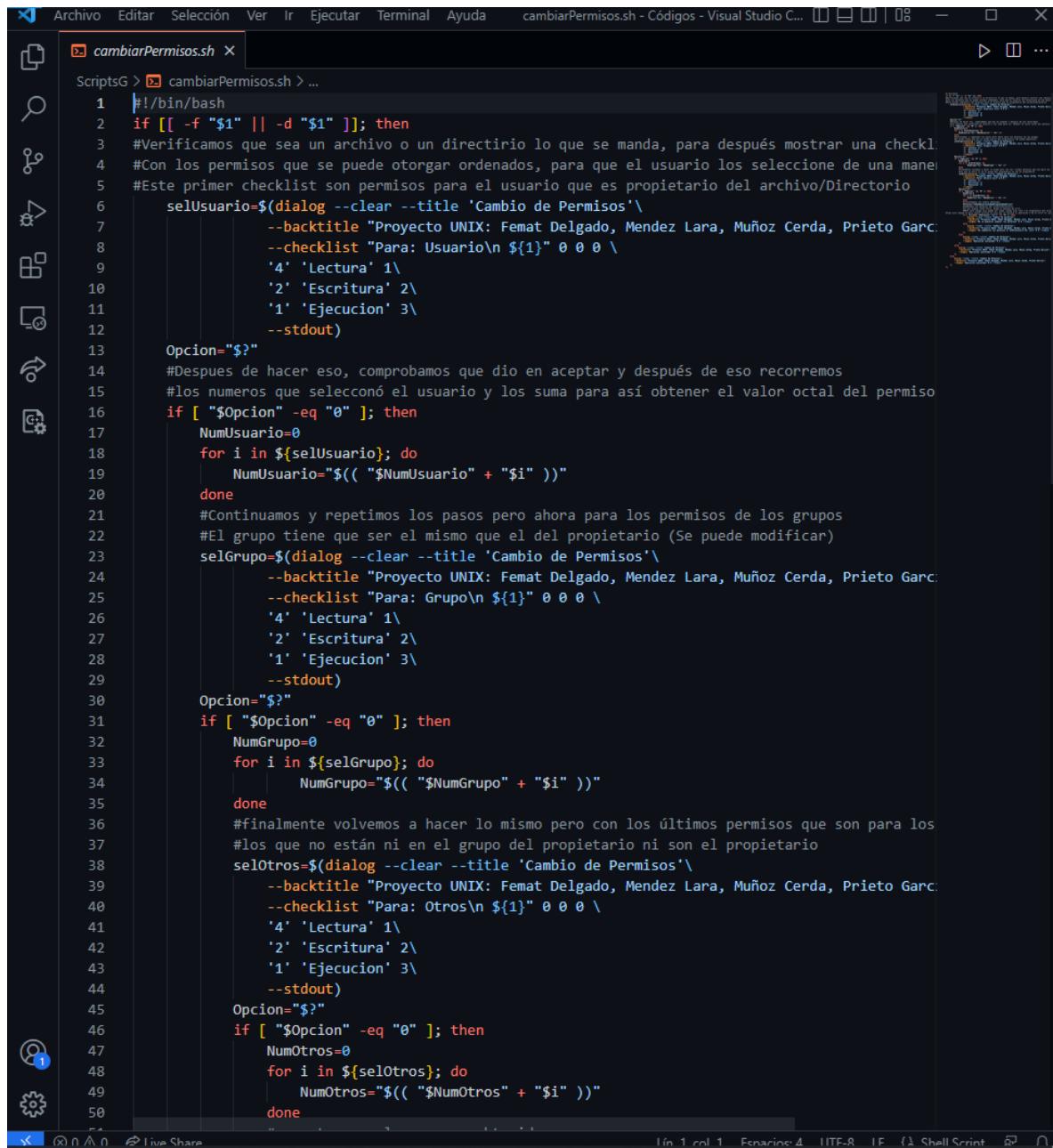


```

cambiarGrupo.sh
ScriptsG > cambiarGrupo.sh > ...
1  #!/bin/bash
2  if [ -d "$1" ] || [ -f "$1" ]; then #Como son Scripts que aplican para casos de directorios o archivos
3      #Generamos un inputbox para ingresar el nombre del nuevo grupo que se quiere hacer el cambio
4      #el archivo o directorio
5      Grupo=$(dialog --clear --title "Selecciono cambiar grupo de un archivo o directorio" \
6          --backtitle "Proyecto UNIX: Femat Delgado, Mendez Lara, Muñoz Cerda, Prieto Garcia" \
7          --inputbox "Ingrese el nombre del nuevo grupo" 14 58 \
8          --stdout)
9  #Usamos grep para buscar el grupo ingresado en el inputbox.
10 #leemos el archivo group en etc donde se guardan los grupos disponibles.
11 #con cut leemos solo la primera posición de la división y como los grupos tienen información separada por :
12 #guardada, por lo que la primera posición que es lo que nos importa, la comparamos con lo que queremos
13 #el archivo
14 if [ -z "$( cat /etc/group | cut -d ":" -f1 | grep -w "$Grupo" )" ]; then
15     dialog --clear --title "Cambiar grupo" \
16         --backtitle "Proyecto UNIX: Femat Delgado, Mendez Lara, Muñoz Cerda, Prieto Garcia" \
17         --msgbox "El grupo ingresado no existe" 0 0 --stdout
18 else
19     #Despues comprobamos que el usuario no sea root y que además está en el grupo de destino
20     #de los dos mencionados, no se puede realizar el comando
21     if [ [ $(whoami) != 'root' && -z "$groups $(groups $(whoami)) | grep \"$Grupo\"" ] ]; then
22         dialog --clear --title "Cambiar grupo" \
23             --backtitle "Proyecto UNIX: Femat Delgado, Mendez Lara, Muñoz Cerda, Prieto Garcia" \
24             --msgbox "No se puede cambiar al grupo, ya que no pertenece a el" 0 0 --stdout
25     else
26         #si es root o pertenece al grupo de destino, hace el cambio de grupo, chown puede hacerlo
27         #cuando se le antepone al nombre ":" con eso chown identifica que es un grupo lo que se indica
28         chown ":${Grupo}" "$1"
29         dialog --clear --title "Cambiar grupo" \
30             --backtitle "Proyecto UNIX: Femat Delgado, Mendez Lara, Muñoz Cerda, Prieto Garcia" \
31             --msgbox "$1 se cambio el grupo a: $Grupo" 0 0 --stdout
32     fi
33 fi
34 fi
35 fi

```

- **cambiarPermisos:** Se verifica que sea un archivo o un directorio lo que se solicita, para después mostrar en un checklist los permisos que se puede otorgar de forma ordenada, para que el usuario los pueda seleccionar.



The screenshot shows a Visual Studio Code interface with a terminal window open. The terminal window has a title bar "cambiarPermisos.sh - Códigos - Visual Studio C..." and a status bar at the bottom showing "Line 1 col 1 Scenario: 4 ITF-R IF { Shell Script }". The main area of the terminal contains the following shell script code:

```

#!/bin/bash
if [ ! -f "$1" || -d "$1" ]; then
    #Verificamos que sea un archivo o un directorio lo que se manda, para después mostrar una checklist
    #Con los permisos que se puede otorgar ordenados, para que el usuario los seleccione de una manera mas sencilla
    #Este primer checklist son permisos para el usuario que es propietario del archivo/Directorio
    selUsuario=$(dialog --clear --title 'Cambio de Permisos' \
        --backtitle "Proyecto UNIX: Femat Delgado, Mendez Lara, Muñoz Cerda, Prieto Garcia" \
        --checklist "Para: Usuario\n${1}" 0 0 0 \
        '4' 'Lectura' 1\
        '2' 'Escritura' 2\
        '1' 'Ejecucion' 3\
        --stdout)
    Opcion="$?"
    #Después de hacer eso, comprobamos que dio en aceptar y después de eso recorremos
    #los numeros que seleccionó el usuario y los suma para así obtener el valor octal del permiso
    if [ "$Opcion" -eq "0" ]; then
        NumUsuario=0
        for i in ${selUsuario}; do
            NumUsuario=$(( ${NumUsuario} + ${i} ))
        done
        #Continuamos y repetimos los pasos pero ahora para los permisos de los grupos
        #El grupo tiene que ser el mismo que el del propietario (Se puede modificar)
        selGrupo=$(dialog --clear --title 'Cambio de Permisos' \
            --backtitle "Proyecto UNIX: Femat Delgado, Mendez Lara, Muñoz Cerda, Prieto Garcia" \
            --checklist "Para: Grupo\n${1}" 0 0 0 \
            '4' 'Lectura' 1\
            '2' 'Escritura' 2\
            '1' 'Ejecucion' 3\
            --stdout)
        Opcion="$?"
        if [ "$Opcion" -eq "0" ]; then
            NumGrupo=0
            for i in ${selGrupo}; do
                NumGrupo=$(( ${NumGrupo} + ${i} ))
            done
            #finalmente volvemos a hacer lo mismo pero con los últimos permisos que son para los
            #los que no están ni en el grupo del propietario ni son el propietario
            selOtros=$(dialog --clear --title 'Cambio de Permisos' \
                --backtitle "Proyecto UNIX: Femat Delgado, Mendez Lara, Muñoz Cerda, Prieto Garcia" \
                --checklist "Para: Otros\n${1}" 0 0 0 \
                '4' 'Lectura' 1\
                '2' 'Escritura' 2\
                '1' 'Ejecucion' 3\
                --stdout)
            Opcion="$?"
            if [ "$Opcion" -eq "0" ]; then
                NumOtros=0
                for i in ${selOtros}; do
                    NumOtros=$(( ${NumOtros} + ${i} ))
                done
            fi
        fi
    fi
fi

```

```

54             #si por alguna razón falla, manda un mensaje de error
55             #NOTA este código solo puede ser usado cuando eres root o el propietario del archivo
56     #Todo este código es para lanzar errores en caso que se cancele la operación o de un error el código
57     if [ "$(chmod "$Permisos" "$1)" -ne "0" ]; then
58         dialog --clear --title 'Cambio de permisos'\
59             --backtitle "Proyecto UNIX: Femat Delgado, Mendez Lara, Muñoz Cerda, Prieto Garcia" \
60             --msgbox "No se pudieron cambiar los permisos" 0 0 --stdout
61     else
62         dialog --clear --title 'Cambio de permisos'\
63             --backtitle "Proyecto UNIX: Femat Delgado, Mendez Lara, Muñoz Cerda, Prieto Garcia" \
64             --msgbox "Se cambiaron los permisos a: ${Permisos}\nEn: ${1}" 0 0 --stdout
65     fi
66     else
67         dialog --clear --title 'Cambio de Permisos'\
68             --backtitle "Proyecto UNIX: Femat Delgado, Mendez Lara, Muñoz Cerda, Prieto Garcia" \
69             --msgbox "Operacion cancelada" 0 0 --stdout
70     fi
71     else
72         dialog --clear --title 'Cambio de Permisos'\
73             --backtitle "Proyecto UNIX: Femat Delgado, Mendez Lara, Muñoz Cerda, Prieto Garcia" \
74             --msgbox "Operacion cancelada" 0 0 --stdout
75     fi
76     else
77         dialog --clear --title 'Cambio de Permisos'\
78             --backtitle "Proyecto UNIX: Femat Delgado, Mendez Lara, Muñoz Cerda, Prieto Garcia" \
79             --msgbox "Operacion cancelada" 0 0 --stdout
80     fi
81 fi

```

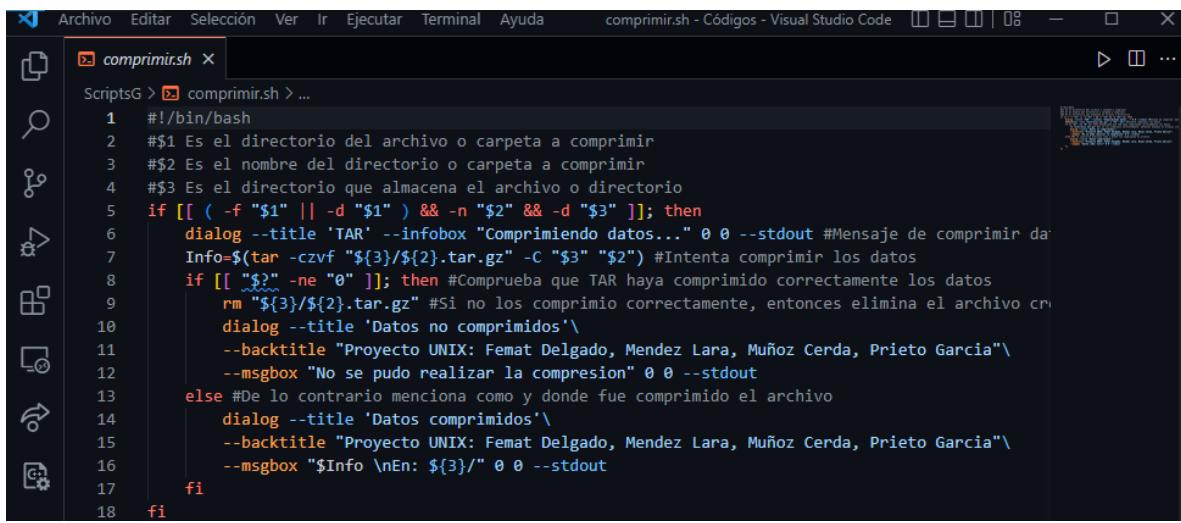
- **cambiarPropietario:** Manda un mensaje preguntando al usuario con que usuario va a cambiar. Comprueba que exista el usuario con el que se quiera cambiar y de verificarlo, lo hace. En caso de que no exista, manda un mensaje de error.

```

#!/bin/bash
#Comprueba que los parametros sean un directorio o archivo
if [ -d "$1" ] || [ -f "$1" ]; then
    #Saca un dialog preguntando al usuario a que usuario quiere transferir la propiedad del archivo
    Usuario=$(dialog --clear --title "Selecciona cambiar propietario de un archivo o directorio" \
                    --backtitle "Proyecto UNIX: Femat Delgado, Mendez Lara, Muñoz Cerda, Prieto Garcia" \
                    --inputbox "Ingrese el nombre del nuevo propietario" 14 58 \
                    --stdout)
    #Busca en el etc/passwd si existe un usuario con ese nombre, entonces permite transferirlo
    if [ -z "$( cat /etc/passwd | cut -d ":" -f1 | grep -w "$Usuario" )" ]; then
        #Si no existe entonces lanza un mensaje de error
        dialog --clear --title "Cambiar propietario" \
                --backtitle "Proyecto UNIX: Femat Delgado, Mendez Lara, Muñoz Cerda, Prieto Garcia" \
                --msgbox "El usuario ingresado no existe" 0 0 --stdout
    else
        #Si existe entonces realiza el cambio y se lo confirma al usuario
        chown "$Usuario" "$1"
        dialog --clear --title "Cambiar propietario" \
                --backtitle "Proyecto UNIX: Femat Delgado, Mendez Lara, Muñoz Cerda, Prieto Garcia" \
                --msgbox "$1 Se cambio el usuario a: $Usuario" 0 0 --stdout
    fi
fi
#NOTA: Este script sólo funciona con la cuenta de root, ya que no se permite cambiar
#el propietario de un archivo o directorio a cuentas normales

```

- **Comprimir:** Comprime un archivo solicitado y muestra en pantalla dónde y cómo fue comprimido. En caso de que falle al comprimir, manda un mensaje de error.

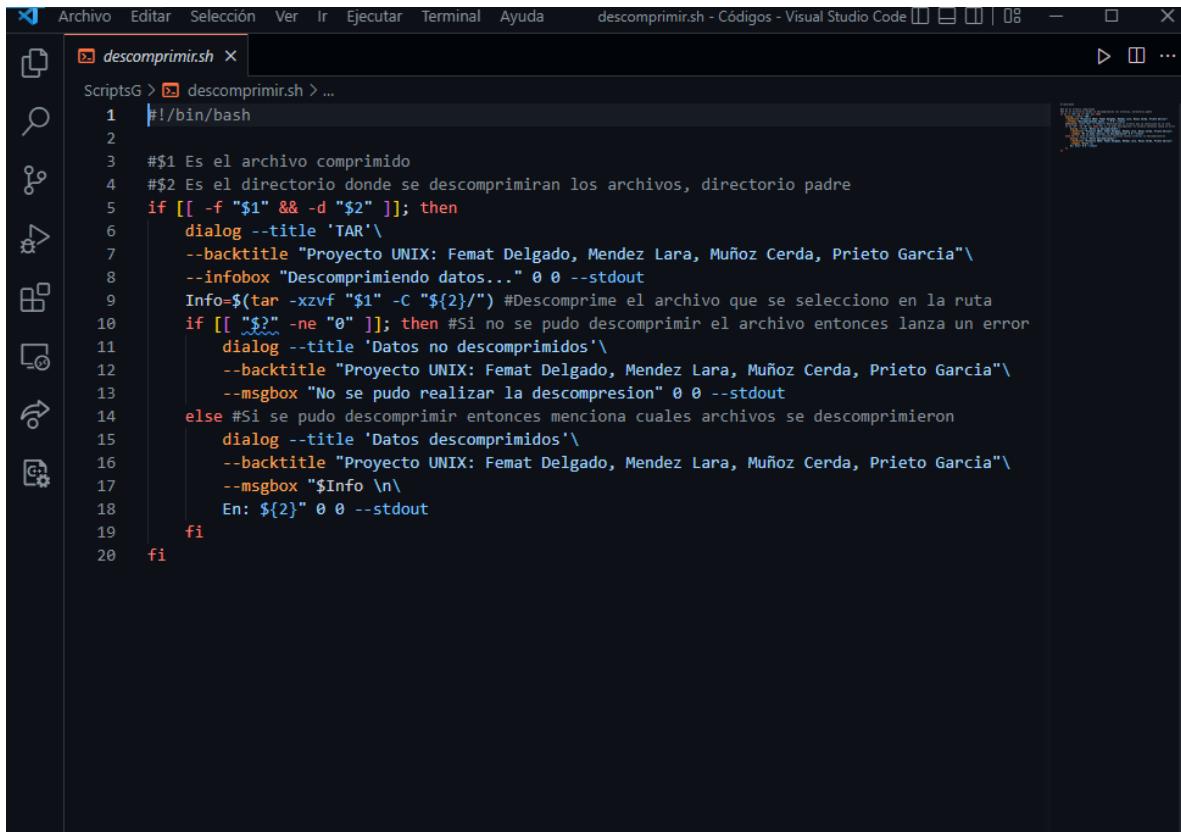


```

1 #!/bin/bash
2 #${1} Es el directorio del archivo o carpeta a comprimir
3 #${2} Es el nombre del directorio o carpeta a comprimir
4 #${3} Es el directorio que almacena el archivo o directorio
5  if [[ ( -f "$1" || -d "$1" ) && -n "$2" && -d "$3" ]]; then
6      dialog --title 'TAR' --infobox "Comprimiendo datos..." 0 0 --stdout #Mensaje de comprimir da
7      Info=$(tar -czvf "${3}/${2}.tar.gz" -C "$3" "$2") #Intenta comprimir los datos
8      if [[ $? -ne "0" ]]; then #Comprueba que TAR haya comprimido correctamente los datos
9          rm "${3}/${2}.tar.gz" #Si no los comprimio correctamente, entonces elimina el archivo cr
10         dialog --title 'Datos no comprimidos'\n11         --backtitle "Proyecto UNIX: Femat Delgado, Mendez Lara, Muñoz Cerda, Prieto Garcia"\n12         --msgbox "No se pudo realizar la compresion" 0 0 --stdout
13     else #De lo contrario menciona como y donde fue comprimido el archivo
14         dialog --title 'Datos comprimidos'\n15         --backtitle "Proyecto UNIX: Femat Delgado, Mendez Lara, Muñoz Cerda, Prieto Garcia"\n16         --msgbox "$Info \nEn: ${3}/" 0 0 --stdout
17     fi
18 fi

```

- **Descomprimir:** Descomprime el/los archivo(s) que se seleccione desde la ruta solicitada, mostrando cuales fueron los archivos descomprimidos, en caso de fallar manda un mensaje de error.

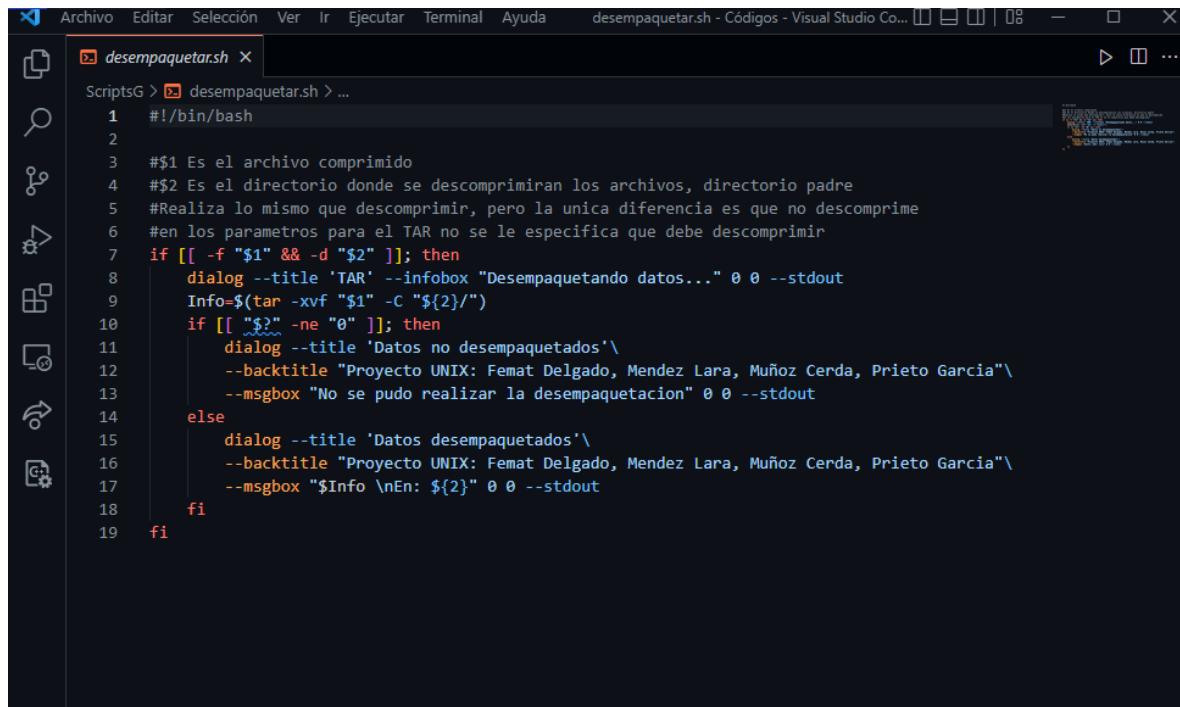


```

1 #!/bin/bash
2
3 #${1} Es el archivo comprimido
4 #${2} Es el directorio donde se descomprimiran los archivos, directorio padre
5  if [[ -f "$1" && -d "$2" ]]; then
6      dialog --title 'TAR'\n7      --backtitle "Proyecto UNIX: Femat Delgado, Mendez Lara, Muñoz Cerda, Prieto Garcia"\n8      --infobox "Descomprimiendo datos..." 0 0 --stdout
9      Info=$(tar -xvf "$1" -C "${2}") #Descomprime el archivo que se selecciono en la ruta
10     if [[ $? -ne "0" ]]; then #Si no se pudo descomprimir el archivo entonces lanza un error
11         dialog --title 'Datos no descomprimidos'\n12         --backtitle "Proyecto UNIX: Femat Delgado, Mendez Lara, Muñoz Cerda, Prieto Garcia"\n13         --msgbox "No se pudo realizar la descompresion" 0 0 --stdout
14     else #Si se pudo descomprimir entonces menciona cuales archivos se descomprimieron
15         dialog --title 'Datos descomprimidos'\n16         --backtitle "Proyecto UNIX: Femat Delgado, Mendez Lara, Muñoz Cerda, Prieto Garcia"\n17         --msgbox "$Info \n\nEn: ${2}" 0 0 --stdout
18     fi
19 fi

```

- **Desempaquetar:** A los .TAR les saca los archivos que tienen contenidos.



```

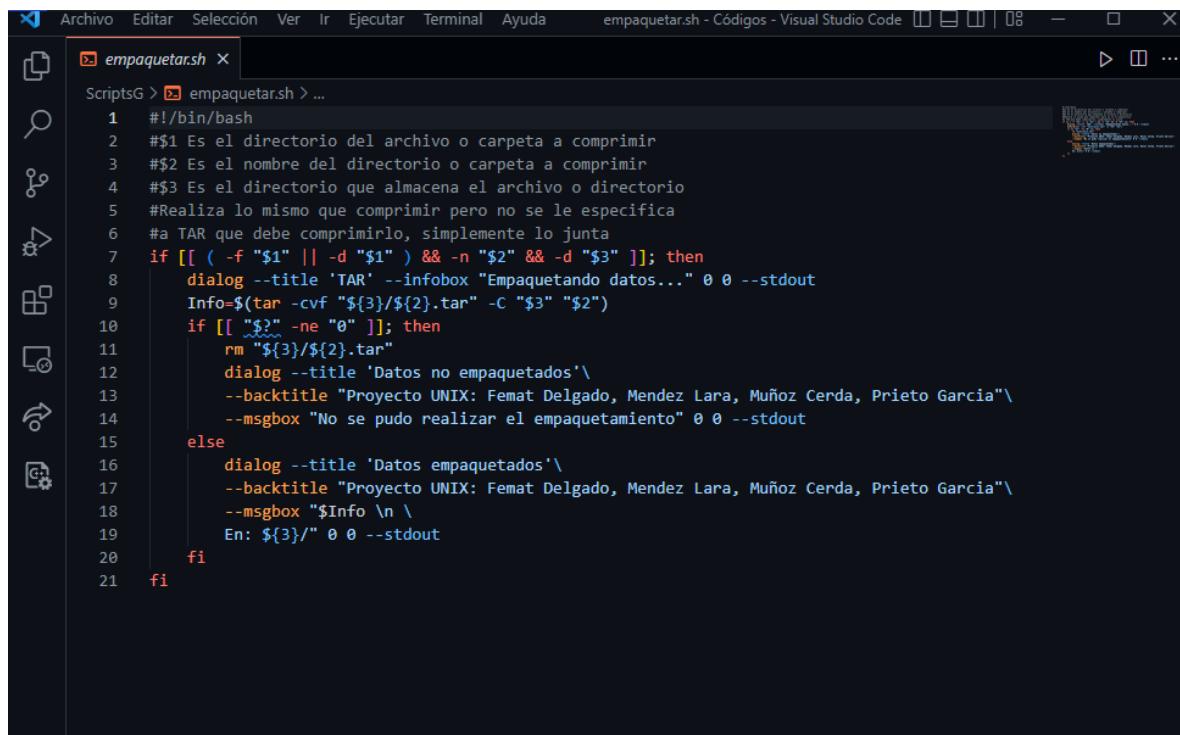
1 #!/bin/bash
2
3 #\$1 Es el archivo comprimido
4 #\$2 Es el directorio donde se descomprimiran los archivos, directorio padre
5 #Realiza lo mismo que descomprimir, pero la unica diferencia es que no descomprime
6 #en los parametros para el TAR no se le especifica que debe descomprimir
7 if [[ -f "$1" && -d "$2" ]]; then
8     dialog --title 'TAR' --infobox "Desempaquetando datos..." 0 0 --stdout
9     Info=$(tar -xvf "$1" -C "${2}/")
10    if [[ $? -ne "0" ]]; then
11        dialog --title 'Datos no desempaquetados'\
12            --backtitle "Proyecto UNIX: Femat Delgado, Mendez Lara, Muñoz Cerdá, Prieto García"\  

13            --msgbox "No se pudo realizar la desempaquetación" 0 0 --stdout
14    else
15        dialog --title 'Datos desempaquetados'\
16            --backtitle "Proyecto UNIX: Femat Delgado, Mendez Lara, Muñoz Cerdá, Prieto García"\  

17            --msgbox "$Info \nEn: ${2}" 0 0 --stdout
18    fi
19 fi

```

- **Empaquetar:** Junta los archivos, pero no los comprime, como lo hace el comprimir.



```

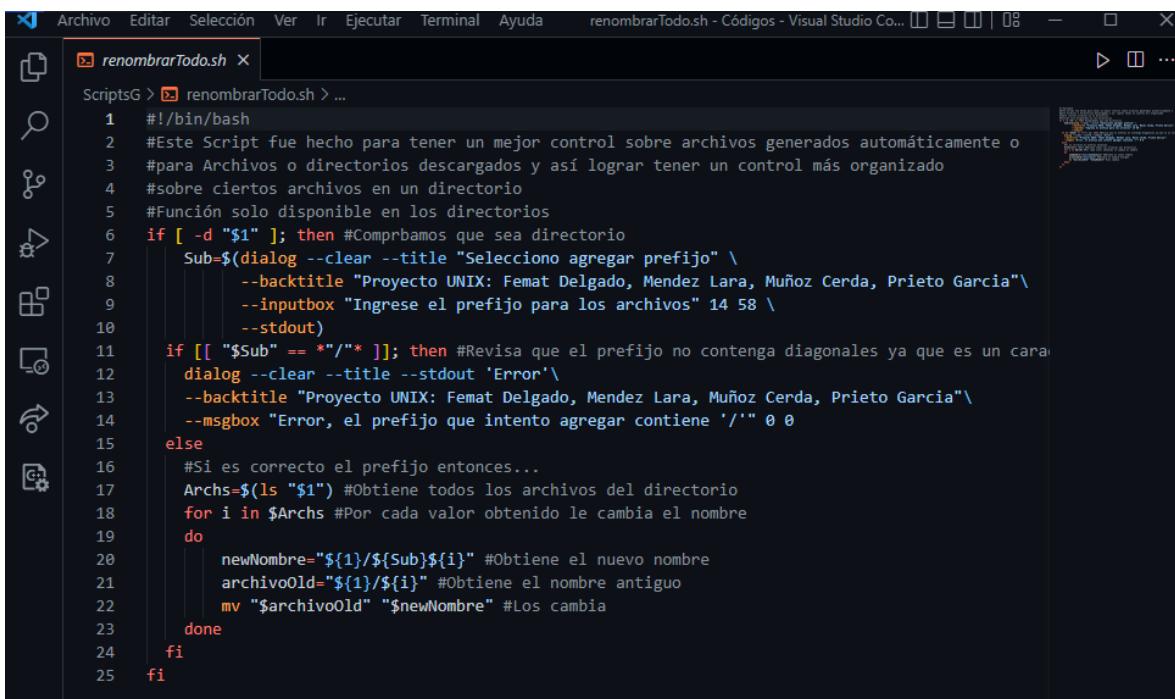
1 #!/bin/bash
2 #\$1 Es el directorio del archivo o carpeta a comprimir
3 #\$2 Es el nombre del directorio o carpeta a comprimir
4 #\$3 Es el directorio que almacena el archivo o directorio
5 #Realiza lo mismo que comprimir pero no se le especifica
6 #a TAR que debe comprimirlo, simplemente lo junta
7 if [[ ( -f "$1" || -d "$1" ) && -n "$2" && -d "$3" ]]; then
8     dialog --title 'TAR' --infobox "Empaquetando datos..." 0 0 --stdout
9     Info=$(tar -cvf "${3}/${2}.tar" -C "$3" "$2")
10    if [[ $? -ne "0" ]]; then
11        rm "${3}/${2}.tar"
12        dialog --title 'Datos no empaquetados'\
13            --backtitle "Proyecto UNIX: Femat Delgado, Mendez Lara, Muñoz Cerdá, Prieto García"\  

14            --msgbox "No se pudo realizar el empaquetamiento" 0 0 --stdout
15    else
16        dialog --title 'Datos empaquetados'\
17            --backtitle "Proyecto UNIX: Femat Delgado, Mendez Lara, Muñoz Cerdá, Prieto García"\  

18            --msgbox "$Info \n \n En: ${3}/" 0 0 --stdout
19    fi
20 fi

```

- **renombrarTodo:** Tiene la funcionalidad de tener un mejor control sobre archivos generados automáticamente o para archivos/directorios descargados, para así lograr tener un control más organizado sobre archivos específicos en un directorio. Esta función esta solo disponible en los directorios.



The screenshot shows a Visual Studio Code window with the following details:

- Title Bar:** Archivo, Editar, Selección, Ver, Ir, Ejecutar, Terminal, Ayuda, renombrarTodo.sh - Códigos - Visual Studio Co... (with icons for file, search, copy, etc.)
- File List:** ScriptsG > renombrarTodo.sh > ...
- Code Editor:**

```

1  #!/bin/bash
2  #Este Script fue hecho para tener un mejor control sobre archivos generados automáticamente o
3  #para Archivos o directorios descargados y así lograr tener un control más organizado
4  #sobre ciertos archivos en un directorio
5  #Función solo disponible en los directorios
6  if [ -d "$1" ]; then #Comprobamos que sea directorio
7      Sub=$(dialog --clear --title "Selecciono agregar prefijo" \
8          --backtitle "Proyecto UNIX: Femat Delgado, Méndez Lara, Muñoz Cerda, Prieto García" \
9          --inputbox "Ingrese el prefijo para los archivos" 14 58 \
10         --stdout)
11  if [[ "$Sub" == *"/"* ]]; then #Revisa que el prefijo no contenga diagonales ya que es un carac
12      dialog --clear --title --stdout 'Error' \
13          --backtitle "Proyecto UNIX: Femat Delgado, Méndez Lara, Muñoz Cerda, Prieto García" \
14          --msgbox "Error, el prefijo que intento agregar contiene '/' 0 0
15 else
16     #Si es correcto el prefijo entonces...
17     Archs=$(ls "$1") #Obtiene todos los archivos del directorio
18     for i in $Archs #Por cada valor obtenido le cambia el nombre
19     do
20         newNombre="${i}/${Sub}${i}" #Obtiene el nuevo nombre
21         archivoOld="${i}/${i}" #Obtiene el nombre antiguo
22         mv "$archivoOld" "$newNombre" #Los cambia
23     done
24   fi
25 fi

```

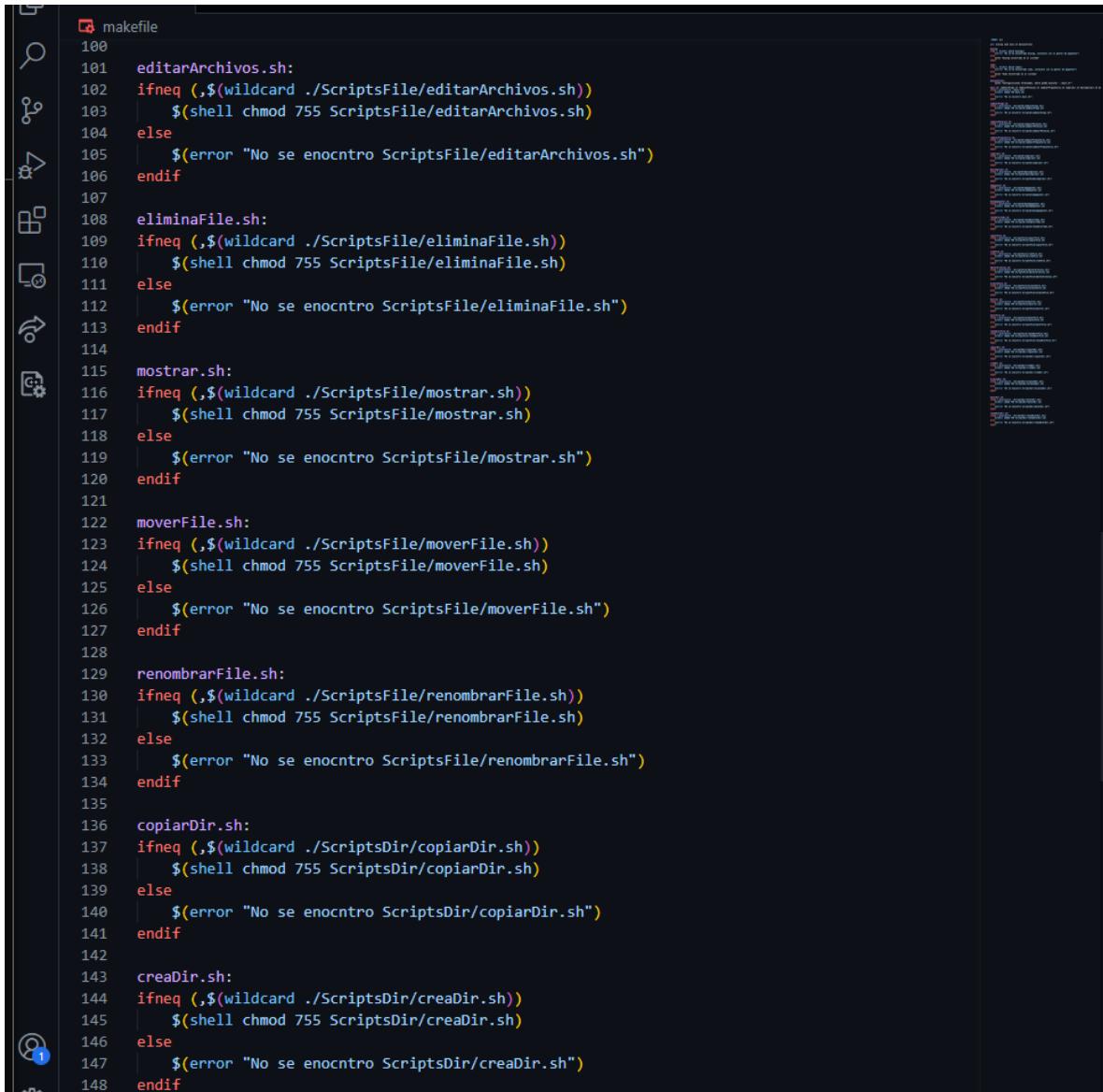
- **Makefile:** Revisa que estén los programas necesarios, los scripts y modifica los permisos para que se puedan ejecutar.

```
makefile - Códigos - Visual Studio Code
Archivo Editar Selección Ver Ir Ejecutar Terminal Ayuda
makefile • 108 - □ □ □ ...
```

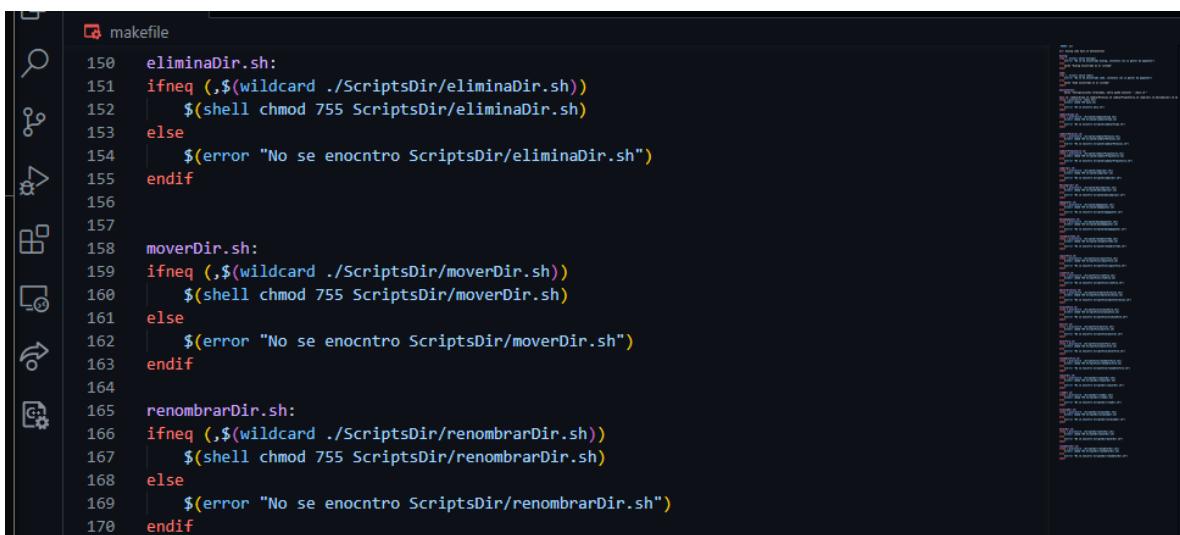
```
makefile
1 |.PHONY: all
2 |
3 |all: dialog sudo main.sh mensajefinal
4 |
5 |dialog:
6 |ifeq (, $(shell which dialog))
7 |    $(error "No se ha encontrado dialog, instalelo con su gestor de paquetes")
8 |else
9 |    @echo "Dialog encontrado en el sistema"
10|endif
11|
12|sudo:
13|ifeq (, $(shell which sudo))
14|    $(error "No se ha encontrado sudo, instalelo con su gestor de paquetes")
15|else
16|    @echo "Sudo encontrado en el sistema"
17|endif
18|
19|mensajefinal:
20|    @echo "Configuraciones terminadas, ahora puede ejecutar './main.sh'"
21|
22|main.sh: cambiarGrupo.sh cambiarPermisos.sh cambiarPropietario.sh comprimir.sh descomprimir.sh e
23|ifeq (,$(wildcard ./main.sh))
24|    $(shell chmod 755 main.sh)
25|else
26|    $(error "No se encontro main.sh")
27|endif
28|
29|cambiarGrupo.sh:
30|ifeq (,$(wildcard ./ScriptsG/cambiarGrupo.sh))
31|    $(shell chmod 755 ScriptsG/cambiarGrupo.sh)
32|else
33|    $(error "No se encontro ScriptsG/cambiarGrupo.sh")
34|endif
35|
36|
37|cambiarPermisos.sh:
38|ifeq (,$(wildcard ./ScriptsG/cambiarPermisos.sh))
39|    $(shell chmod 755 ScriptsG/cambiarPermisos.sh)
40|else
41|    $(error "No se enocntro ScriptsG/cambiarPermisos.sh")
42|endif
43|
44|cambiarPropietario.sh:
45|ifeq (,$(wildcard ./ScriptsG/cambiarPropietario.sh))
46|    $(shell chmod 755 ScriptsG/cambiarPropietario.sh)
47|else
48|    $(error "No se enocntró ScriptsG/cambiarPropietario.sh")
49|endif
50|
```

The screenshot shows a Visual Studio Code window with a dark theme. The main area displays a Makefile named 'makefile'. The code contains several shell script functions: 'comprimir.sh', 'descomprimir.sh', 'empaquetar.sh', 'desempaquetar.sh', 'renombrarTodo.sh', 'copiarFile.sh', and 'creaFile.sh'. Each function includes logic to check for the existence of a script using a wildcard and to change its permissions to 755 if it exists. Syntax highlighting is used for keywords like 'ifeq' and file paths. A code completion dropdown is visible on the right side of the editor, listing various file names and functions. The bottom status bar shows file navigation, line numbers, and other standard VS Code interface elements.

```
makefile
50
51  comprimir.sh:
52  ifneq (,$(wildcard ./ScriptsG/comprimir.sh))
53  |    $(shell chmod 755 ScriptsG/comprimir.sh)
54  else
55  |    $(error "No se enocntro ScriptsG/comprimir.sh")
56  endif
57
58  descomprimir.sh:
59  ifneq (,$(wildcard ./ScriptsG/descomprimir.sh))
60  |    $(shell chmod 755 ScriptsG/descomprimir.sh)
61  else
62  |    $(error "No se enocntro ScriptsG/descomprimir.sh")
63  endif
64
65  empaquetar.sh:
66  ifneq (,$(wildcard ./ScriptsG/empaquetar.sh))
67  |    $(shell chmod 755 ScriptsG/empaquetar.sh)
68  else
69  |    $(error "No se enocntro ScriptsG/empaquetar.sh")
70  endif
71
72  desempaquetar.sh:
73  ifneq (,$(wildcard ./ScriptsG/desempaquetar.sh))
74  |    $(shell chmod 755 ScriptsG/desempaquetar.sh)
75  else
76  |    $(error "No se enocntro ScriptsG/desempaquetar.sh")
77  endif
78
79  renombrarTodo.sh:
80  ifneq (,$(wildcard ./ScriptsG/renombrarTodo.sh))
81  |    $(shell chmod 755 ScriptsG/renombrarTodo.sh)
82  else
83  |    $(error "No se enocntro ScriptsG/renombrarTodo.sh")
84  endif
85
86
87  copiarFile.sh:
88  ifneq (,$(wildcard ./ScriptsFile/copiarFile.sh))
89  |    $(shell chmod 755 ScriptsFile/copiarFile.sh)
90  else
91  |    $(error "No se enocntro ScriptsFile/copiarFile.sh")
92  endif
93
94  creaFile.sh:
95  ifneq (,$(wildcard ./ScriptsFile/creaFile.sh))
96  |    $(shell chmod 755 ScriptsFile/creaFile.sh)
97  else
98  |    $(error "No se enocntro ScriptsFile/creaFile.sh")
99  endif
100
```



```
makefile
100
101 editarArchivos.sh:
102     ifneq (,$(wildcard ./ScriptsFile/editarArchivos.sh))
103         $(shell chmod 755 ScriptsFile/editarArchivos.sh)
104     else
105         $(error "No se enocntro ScriptsFile/editarArchivos.sh")
106     endif
107
108 eliminaFile.sh:
109     ifneq (,$(wildcard ./ScriptsFile/eliminaFile.sh))
110         $(shell chmod 755 ScriptsFile/eliminaFile.sh)
111     else
112         $(error "No se enocntro ScriptsFile/eliminaFile.sh")
113     endif
114
115 mostrar.sh:
116     ifneq (,$(wildcard ./ScriptsFile/mostrar.sh))
117         $(shell chmod 755 ScriptsFile/mostrar.sh)
118     else
119         $(error "No se enocntro ScriptsFile/mostrar.sh")
120     endif
121
122 moverFile.sh:
123     ifneq (,$(wildcard ./ScriptsFile/moverFile.sh))
124         $(shell chmod 755 ScriptsFile/moverFile.sh)
125     else
126         $(error "No se enocntro ScriptsFile/moverFile.sh")
127     endif
128
129 renombrarFile.sh:
130     ifneq (,$(wildcard ./ScriptsFile/renombrarFile.sh))
131         $(shell chmod 755 ScriptsFile/renombrarFile.sh)
132     else
133         $(error "No se enocntro ScriptsFile/renombrarFile.sh")
134     endif
135
136 copiarDir.sh:
137     ifneq (,$(wildcard ./ScriptsDir/copiarDir.sh))
138         $(shell chmod 755 ScriptsDir/copiarDir.sh)
139     else
140         $(error "No se enocntro ScriptsDir/copiarDir.sh")
141     endif
142
143 creaDir.sh:
144     ifneq (,$(wildcard ./ScriptsDir/creaDir.sh))
145         $(shell chmod 755 ScriptsDir/creaDir.sh)
146     else
147         $(error "No se enocntro ScriptsDir/creaDir.sh")
148     endif
```



```
makefile
150
151     eliminaDir.sh:
152         ifneq (,$(wildcard ./ScriptsDir/eliminaDir.sh))
153             $(shell chmod 755 ScriptsDir/eliminaDir.sh)
154         else
155             $(error "No se enocntro ScriptsDir/eliminaDir.sh")
156         endif
157
158     moverDir.sh:
159         ifneq (,$(wildcard ./ScriptsDir/moverDir.sh))
160             $(shell chmod 755 ScriptsDir/moverDir.sh)
161         else
162             $(error "No se enocntro ScriptsDir/moverDir.sh")
163         endif
164
165     renombrarDir.sh:
166         ifneq (,$(wildcard ./ScriptsDir/renombrarDir.sh))
167             $(shell chmod 755 ScriptsDir/renombrarDir.sh)
168         else
169             $(error "No se enocntro ScriptsDir/renombrarDir.sh")
170         endif
```

Fuentes de consulta:

- <https://conocimientolibre.mx/shell-scripting/>
- <https://sanchezcorbalan.es/que-es-un-shell-script/>
- <https://aplicacionesysistemas.com/en/dialog-crear-menus-tus-scripts/>
- [https://www.ecured.cu/Dialog_\(programa_de_Linux\)](https://www.ecured.cu/Dialog_(programa_de_Linux))
- http://gluc.unicauca.edu.co/index.php/Programaci%F3n_con_Ncurses#:~:text=Las%20ncurses%20son%20librerias%20que,y%20de%20forma%20muy%20fácil
- <https://www.netinbag.com/es/internet/what-is-a-metacharacter.html>