

Escuela de Ingeniería en Computación

Principios de Sistemas Operativos

Proyecto #2

Simple TAR

Integrantes:

Francisco Villanueva Quirós - 2021043887

Jarod Cervantes Gutiérrez - 2019243821

Semestre II

2024

Fecha de Entrega:

28/10/2024

Introducción

Para el segundo proyecto, se nos solicitó desarrollar el proyecto star, este consiste en un empacador de archivos. Esto para hacer un sistema que permita empaquetar múltiples archivos en uno solo, haciendo más eficiente el almacenamiento de archivos y la movilidad de estos. Similar al programa tar, este proyecto se va a ejecutar desde línea de comandos, para esto se va a seguir la estructura del comando tar:

- tar <opciones> <archivoSalida> <archivo1> <archivo2> <archivoN>

Por lo que, hay que implementar en el sistema una serie de funciones para lograr que el sistema permita crear, actualizar, listar, extraer, borrar entre otras funcionalidades que nos permita hacer las operaciones sobre los archivos tar.

Descripción del problema (enunciado)

El programa "star" debe implementar un empacador de archivos capaz de aceptar comandos básicos y realizar operaciones típicas de empaquetado y desempaquetado de archivos en un entorno UNIX y realizado en un programa en C. Para realizar el programa, se pusieron algunos requerimientos para hacer que el funcionamiento sea el deseado, entre ellos están:

- Se deberá utilizar una lista enlazada de bloques, donde cada bloque debe contener el número del siguiente bloque que contiene el archivo.
- El archivo debe almacenarse en bloques de 256kb
- Se va a almacenar el nombres del archivo, tamaño e índice del primer bloque, esto en una estructura de tamaño fijo.
- Cuando un archivo es borrado, los bloques utilizados deben quedar marcados como libres para reutilización sin reducir el tamaño total del archivo empacado.
- Los archivos pueden ser sobrescritos o actualizados con el comando -u sin necesidad de eliminarlos manualmente

- Todas las operaciones de modificación y desfragmentación deben realizarse directamente en el archivo empaado, sin la ayuda de archivos temporales auxiliares.

Definición de estructuras de datos

Para la creación del programa se realizaron diferentes estructuras de datos. Cada una de estas son necesarias para cumplir con los requerimientos mencionados anteriormente.

- fileBlock
 - Estructura que almacena los datos de los archivos que se ingresan
 - Contiene campos:
 - char name[BLOCK_SIZE]
 - int next
 - Funciones:
 - FileBlock *newFileBlock()
 - void setNameFileBlock(FileBlock *fileBlock, const char data[BLOCK_SIZE])
 - void setNextFileBlock(FileBlock *fileBlock, int next)
 - void serializeFileBlock(FileBlock *fileBlock, FILE *file)
 - FileBlock *deserializeFileBlock(FILE *file)
 - int getOffsetFileBlock()
- fileHeader
 - Estructura que almacena la información de un archivo
 - Contiene campos:
 - char name
 - int size
 - int first
 - int last
 - int totalBlocks
 - int index
 - Funciones:
 - FileHeader *newFileHeader()
 - void setNameFileHeader(FileHeader *fileHeader, const char *name)
 - void serializeFileHeader(FileHeader *fileHeader, FILE *file)

- FileHeader *deserializeFileHeader(FILE *file)
 - char *toStringFileHeader(FileHeader *fileHeader)
 - int isFileHeaderAvailable(FileHeader *fileHeader)
- tableFile
 - Estructura que representa el archivo star con los archivos empaquetados
 - Contiene campos:
 - char fileName[FILE_NAME_SIZE]
 - int filesCount
 - int blockCount
 - FileHeader *fileHeader[FILES_NUM]
 - FileHeader *freeBlocksHeader
 - Funciones:
 - TableFile *newTableFile(const char *fileName)
 - void openTableFile(TableFile *tableFile, const char *mode)
 - void addFile(TableFile *tableFile, const char *fileName)
 - int getOffsetTableFile(TableFile *tableFile)
 - FileHeader *getFileHeaderToUse(TableFile *tableFile)
 - int getBlockAvailable(TableFile *tableFile)
 - FileHeader *getFileHeader(TableFile *tableFile, const char *fileName)
 - TableFile *loadTableFile(char *inputFile)
 - void extractAllFiles(TableFile *tableFile, const char *outputDirectory)
 - void serializeTableFile(TableFile *table, const char *filename)
 - TableFile *deserializeTableFile(const char *filename)
 - void create(TableFile *tableFile, const char *outputFile)
 - void delete(TableFile *tableFile, const char *fileName)
 - void update(TableFile *tableFile, const char *fileName)
 - void listFiles(TableFile *tableFile)

Descripción detallada y explicación de componentes principales

A. Mecanismo de acceso a archivos

Para el mecanismo del sistema de archivos, se trabajó con el archivos .tar y con los archivos que se desean agregar al tar, serían los archivos que se buscan empaquetar. Para la escritura en el archivo tar, se crea la función `addFile` que recibe una declaración del `tableFile` y un nombre del archivo que se desea agregar al .tar. Antes de la escritura en el tar, se hacen las validación de verificar que el archivo no esté ya ingresado y que haya espacio disponible. Posteriormente, se establecen los headers correspondientes y mediante la siguiente línea de comando se abre el archivo: `"FILE *star = fopen(tableFile->fileName, "rb+");"`, es necesario calcular el offset para determinar los saltos que debe dar el archivo, ya que los bloques no necesariamente van seguidos. Luego, es necesario buscar los bloques libres disponibles en donde se estaría agregando el nuevo archivo que estoy cargando. Un paso importante es escribir en los bloques correspondientes, esto se hace calculando el offset que tiene el archivo y mediante el comando `fseek`, se puede hacer saltos y escribir en los bloques debidos.

Otro de los mecanismos para el acceso de archivos en el tar y realizar operaciones se logra mediante la función `delete`. Esta es la encargada de eliminar un archivo del empaquetado y marcar los bloques que se eliminaron como bloques libres. Esto se logra buscando el archivo por su header y marcándolo como eliminado, haciendo que se sume el espacio disponible en el `table file`. Esto permite que se puedan volver a utilizar estos bloques que ya están libres. Es necesario, debido a cómo se maneja la estructura, que el último bloque del archivo utilizado apunte al primer bloque disponible.

El siguiente mecanismo igual nos permite acceder a archivos tar a realizar operaciones. La función `update` nos permite sobrescribir archivos que ya fueron cargados en el tar. Esto se hace cargando dos archivos, el archivo que se busca obtener la información para actualizar y el archivo tar. Esto se logra haciendo uso de las funciones `open` y `write` en los diferentes archivos. Se realiza un ciclo que

lee los datos del archivo y los escribe en el empaquetado. En este se debe validar que el update que se va a hacer en el archivo cabe en el bloque que se encuentra, de no ser así se debe asignar un bloque nuevo. Luego la función debe actualizar los datos del header, agregar el nuevo tamaño de archivo y la nueva posición del último bloque utilizado.

La función de `extractAllFiles`, se encarga de extraer todos los archivos y almacenarlos en un directorio de salida que se especifica. Para iniciar, primero se debe apuntar al offset indicado en los headers que contiene el empaquetado. La función va bloque por bloque, esto se logra ya que los bloques apuntan al siguiente bloque y de esta forma se conecta con los archivos que se van a extraer. Se utiliza la función `deserialize` que se va a comentar más adelante, para obtener toda la información del bloque. Mediante una constante de bytes restantes, se puede hacer la extracción completa del contenido y que no se detenga hasta que ya no haya información restante.

Por último las funciones generales de `serialize` y `deserialize`. Estas funciones se encargan de leer y escribir el `tableFile` en un archivo binario. Primeramente la función `serialize`, esta recibe una instancia de `TableFile` y un nombre de archivo de salida. Se utiliza la función `write` para escribir el nombre del archivo. Mediante un bucle se guarda la información de los datos del header en los bloques libres y de igual forma se usa el `write` para almacenar todos estos datos. Por otra parte, la función `deserialize`. Se inicia creando un estructura `tableFile` nueva y vacía. Mediante el comando `read`, se lee el nombre del archivo y guardarlos en el campo del header en el `tableFile`. Luego mediante un bucle, lee un bloque de encabezado desde el archivo y lo convierte en un `file header`, haciendo un archivo en específico. Luego, se asigna la información a los bloques libres.

B. Estructura de directorios internos

Para la estructura de directorios, se siguió los requisitos del proyecto. Primeramente, se utiliza la estructura del `tableFile`, esta es la que guarda la información del archivo empaquetado, como el nombres, cantidad de archivos y la cantidad de bloques, además, de manejar la cantidad de bloques disponibles que gestiona la utilización de estos.

Posteriormente, ya directamente con el contenido del star, se creó la estructura de `Fileheader`. Cada archivo contiene un header que almacena datos importante para las operaciones como: nombre, tamaño, los bloques utilizados y punteros hacia los siguientes bloques donde está el contenido del archivo.

Por último, la estructura de bloques. Para esta se creó la estructura de `FileBlock`, el contenido de los archivos que se cargan al empaquetado es almacenado en bloques. Cada uno de los bloques contiene una parte del archivo y un puntero hacia el siguiente bloque que corresponde con la información o datos restantes de este archivo. Esto se logra enlazando los bloques con punteros, lo que hace que la gestión de estos sea más fácil.

C. Estrategia de manejo de bloques libres

Como se mostró anteriormente en la estructura **TableFile**, se tiene un variable `freeBlocksHeader` de tipo **FileHeader**, donde se indican cuál es el primer(first) y último(last) bloque libre. La idea de está estructura es buscar en disco la posición del primer bloque, cuando sea distinto de -1, e ir saltando de bloque en bloque para solicitar bloques libres.

Cuando se crea el archivo vía la opción **-c** no existen bloques libres, es hasta que se eliminan o actualiza un archivo a un menor tamaño que se crean los bloques libres. Al eliminar un primer archivo lo único que se hace es asignar en los campos primer(first) y último(last) de **freeBlocksHeader** los valores del archivo eliminado. Y cuando ya se a eliminado un archivo previamente, se va a la posición de **freeBlocksHeader->last** para encontrar el bloque que indica y así cambiar su valor de -1 al primer bloque del archivo borrado, y luego el valor de

freeBlocksHeader->last pasa a ser el último del archivo borrado.

Por último cada vez que se agregan bloques a **freeBlocksHeader** se ordena la lista con el fin de usar siempre los bloques de menor denominación y facilitar también el proceso de desfragmentación.

D. Proceso de desfragmentación del archivo

El proceso de desfragmentación consiste en los siguientes pasos:

1. Obtener una lista de bloques libres.
 - a. Esta lista ya está ordenada de menor a mayor.
 - b. Si no existen bloques libres no se debe de desfragmentar.
2. Se itera sobre cada archivo válido.
3. Se itera sobre cada bloque del archivo.
4. Se compara cada bloque del archivo con el primer bloque libre, si el bloque del archivo es mayor significa que se tiene que mover a la posición del bloque libre.
 - a. Si se mueve, se escribe en disco la nueva posición del bloque con los datos.
 - b. Luego se actualiza el bloque previo del archivo con la nueva posición del bloque movido
5. Se libera el bloque antiguo y se añade a bloques libres, además que se mueven los **k+1** bloques a **k** hasta que el bloque antiguo quede ordenado en la lista, esta lista siempre se mantiene ordenada.
6. Una vez procesados todos los bloques del archivo se actualiza su **FileHeader** en disco con su nueva posición del primer y el último bloque.
7. Una vez se haya iterado sobre todos los archivos, los únicos bloques libres son los de mayor índice. Los del final del archivo. Y se resetea a los valores

iniciales de **freeBlocksHeader** como si no existieran bloques libres. Esta nueva información se escribe en disco.

8. Una vez actualizada la información en disco. Se trunca el archivo dejando por fuera los **n** bloques libres.
9. Se guarda el archivo.

Análisis de resultados de pruebas

Estas pruebas se manejaron con bloques de 64 bytes, y un máximo de 5 archivos. Sin embargo el programa permite bloques de 256kb y hasta 250 archivos.

1. Crear archivo:

Se hace una prueba, cargando 4 archivos de texto y realizando el empaquetado mediante el star:

```
(base) franvq09@MBPdeFrancisco S-tar % ./star -vvvcf test.star archivo1.txt archivo2.txt nuevoArchivo.txt otro.txt  
argc: 7
```



1.1 Se realiza la prueba de crear el archivo, con archivos de diferentes extensiones. Como se puede observar, hay archivos .txt .pdf y .c

```
(base) franvq09@MBPdeFrancisco S-tar % ./star -vvvcf test.star ope13-Colas.pdf fisica.c archivo1.txt archivo2txt otro.txt  
argc: 8
```



2. Update de un archivo:

Se hace un update de los archivos “archivo1.txt” y “archivo2.txt”

```
(base) franvq09@MBPdeFrancisco S-tar % ./star -vvvu test.star archivo1.txt archivo2.txt  
argc: 5  
(base) franvq09@MBPdeFrancisco S-tar %
```

Se puede ver un listado de la información importante que contiene los archivos.

```

• (base) franvq090MBPdeFrancisco S-tar % ./star -vvvv test.star
argc: 3
Archivos en la tabla:
FileHeader: name: 'archivo1.txt', first: 0, last: 0, isDeleted: 0, size: 59, index: 0, blocks: 1.
Blocks : 0
FileHeader: name: 'archivo2.txt', first: 1, last: 1, isDeleted: 0, size: 60, index: 1, blocks: 1.
Blocks : 1
FileHeader: name: 'nuevoArchivo.txt', first: 2, last: 2, isDeleted: 0, size: 21, index: 2, blocks: 1.
Blocks : 2
FileHeader: name: 'otro.txt', first: 3, last: 3, isDeleted: 0, size: 47, index: 3, blocks: 1.
Blocks : 3
FileHeader: name: 'freeBlocksHeader', first: -1, last: -1, isDeleted: 0, size: 0, index: -1, blocks: 0.
Blocks : no blocks

```

2.1 Prueba update realizada con diferentes tipos de extensiones.

```
• (base) franvq09@MBPdeFrancisco S-tar % ./star -vvvvu test.star archivo2.txt
argc: 4
```

Contenido del archivo star

```
(base) Franv@90MHPdeFrancisco 5-tar & ./star -vvvt test.star  
argc: 3  
Archivos en la tabla:  
Fileheader: name:'\0peli3-Colas.pdf' first:0 last: 6990, isDeleted:0, size: 447385, index:0, blocks: 6991.  
Blocks: 0 => 1 => 2 => 3 => 4 => 5 => 6 => 7 => 8 => 9 => 10 => 11 => 12 => 13 => 14 => 15 => 16 => 17 => 18 => 19 => 20 => 21 => 22 => 23 => 24 => 25 => 26 => 27 => 28 => 29 => 30 => 31 =>  
32 => 33 => 34 => 35 => 36 => 37 => 38 => 39 => 40 => 41 => 42 => 43 => 44 => 45 => 46 => 47 => 48 => 49 => 50 => 51 => 52 => 53 => 54 => 55 => 56 => 57 => 58 => 59 => 60 => 61 => 62 => 63 =>  
4 => 65 => 66 => 67 => 68 => 69 => 70 => 71 => 72 => 73 => 74 => 75 => 76 => 77 => 78 => 79 => 80 => 81 => 82 => 83 => 84 => 85 => 86 => 87 => 88 => 89 => 90 => 91 => 92 => 93 => 94 => 95 => 9  
6 => 97 => 98 => 99 => 100 => 101 => 102 => 103 => 104 => 105 => 106 => 107 => 108 => 109 => 110 => 111 => 112 => 113 => 114 => 115 => 116 => 117 => 118 => 119 => 120 => 121 => 122 => 123 =>  
24 => 125 => 126 => 127 => 128 => 129 => 130 => 131 => 132 => 133 => 134 => 135 => 136 => 137 => 138 => 139 => 140 => 141 => 142 => 143 => 144 => 145 => 146 => 147 => 148 => 149 => 150 => 151  
=> 152 => 153 => 154 => 155 => 156 => 157 => 158 => 159 => 160 => 161 => 162 => 163 => 164 => 165 => 166 => 167 => 168 => 169 => 170 => 171 => 172 => 173 => 174 => 175 => 176 => 177 => 178  
179 => 180 => 181 => 182 => 183 => 184 => 185 => 186 => 187 => 188 => 189 => 190 => 191 => 192 => 193 => 194 => 195 => 196 => 197 => 198 => 199 => 200 => 201 => 202 => 203 => 204 => 205 => 206 =>  
207 => 208 => 209 => 210 => 211 => 212 => 213 => 214 => 215 => 216 => 217 => 218 => 219 => 220 => 221 => 222 => 223 => 224 => 225 => 226 => 227 => 228 => 229 => 230 => 231 => 232 => 233 =>  
234 => 235 => 236 => 237 => 238 => 239 => 240 => 241 => 242 => 243 => 244 => 245 => 246 => 247 => 248 => 249 => 250 => 251 => 252 => 253 => 254 => 255 => 256 => 257 => 258 => 259 => 260 => 261 =>  
1 => 262 => 263 => 264 => 265 => 266 => 267 => 268 => 269 => 270 => 271 => 272 => 273 => 274 => 275 => 276 => 277 => 278 => 279 => 280 => 281 => 282 => 283 => 284 => 285 => 286 => 287 => 288  
=> 289 => 290 => 291 => 292 => 293 => 294 => 295 => 296 => 297 => 298 => 299 => 300 => 301 => 302 => 303 => 304 => 305 => 306 => 307 => 308 => 309 => 310 => 311 => 312 => 313 => 314 => 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 =>
```

2888	2889	2900	2901	2902	2903	2904	2905	2906	2907	2908	2909	2910	2911	2912	2913	2914	2915	2916	2917	2918	2919	2920	2921
2922	2923	2924	2925	2926	2927	2928	2929	2930	2931	2932	2933	2934	2935	2936	2937	2938	2939	2940	2941	2942	2943	2944	2945
2946	2947	2948	2949	2950	2951	2952	2953	2954	2955	2956	2957	2958	2959	2960	2961	2962	2963	2964	2965	2966	2967	2968	2969
2970	2971	2972	2973	2974	2975	2976	2977	2978	2979	2980	2981	2982	2983	2984	2985	2986	2987	2988	2989	2990	2991	2992	2993
2994	2995	2996	2997	2998	2999	3000	3001	3002	3003	3004	3005	3006	3007	3008	3009	3010	3011	3012	3013	3014	3015	3016	3017
3018	3019	3020	3021	3022	3023	3024	3025	3026	3027	3028	3029	3030	3031	3032	3033	3034	3035	3036	3037	3038	3039	3040	3041
3042	3043	3044	3045	3046	3047	3048	3049	3050	3051	3052	3053	3054	3055	3056	3057	3058	3059	3060	3061	3062	3063	3064	3065
3066	3067	3068	3069	3070	3071	3072	3073	3074	3075	3076	3077	3078	3079	3080	3081	3082	3083	3084	3085	3086	3087	3088	3089
3090	3091	3092	3093	3094	3095	3096	3097	3098	3099	3100	3101	3102	3103	3104	3105	3106	3107	3108	3109	3110	3111	3112	3113
3114	3115	3116	3117	3118	3119	3120	3121	3122	3123	3124	3125	3126	3127	3128	3129	3130	3131	3132	3133	3134	3135	3136	3137
3138	3139	3140	3141	3142	3143	3144	3145	3146	3147	3148	3149	3150	3151	3152	3153	3154	3155	3156	3157	3158	3159	3160	3161
3162	3163	3164	3165	3166	3167	3168	3169	3170	3171	3172	3173	3174	3175	3176	3177	3178	3179	3180	3181	3182	3183	3184	3185
3186	3187	3188	3189	3190	3191	3192	3193	3194	3195	3196	3197	3198	3199	3200	3201	3202	3203	3204	3205	3206	3207	3208	3209
3210	3211	3212	3213	3214	3215	3216	3217	3218	3219	3220	3221	3222	3223	3224	3225	3226	3227	3228	3229	3230	3231	3232	3233
3234	3235	3236	3237	3238	3239	3240	3241	3242	3243	3244	3245	3246	3247	3248	3249	3250	3251	3252	3253	3254	3255	3256	3257
3258	3259	3260	3261	3262	3263	3264	3265	3266	3267	3268	3269	3270	3271	3272	3273	3274	3275	3276	3277	3278	3279	3280	3281
3282	3283	3284	3285	3286	3287	3288	3289	3290	3291	3292	3293	3294	3295	3296	3297	3298	3299	3300	3301	3302	3303	3304	3305
3306	3307	3308	3309	3310	3311	3312	3313	3314	3315	3316	3317	3318	3319	3320	3321	3322	3323	3324	3325	3326	3327	3328	3329
3330	3331	3332	3333	3334	3335	3336	3337	3338	3339	3340	3341	3342	3343	3344									


```

• (base) franvq09@MBPdeFrancisco S-tar % ./star -vvvvt test.star
argc: 3
Archivos en la tabla:
FileHeader: name: 'archivo1.txt', first: 0, last: 0, isDeleted: 0, size: 59, index: 0, blocks: 1.
Blocks : 0
FileHeader: name: 'nuevoArchivo.txt', first: 2, last: 2, isDeleted: 0, size: 21, index: 2, blocks: 1.
Blocks : 2
FileHeader: name: 'otro.txt', first: 3, last: 3, isDeleted: 0, size: 47, index: 3, blocks: 1.
Blocks : 3
FileHeader: name: 'freeBlocksHeader', first: 1, last: 1, isDeleted: 0, size: 60, index: -1, blocks: 1.
Blocks : 1

```

3.1. Prueba delete con varios archivos de diferentes extensiones:

```

base) Fran99@99MDeFrancisco S-star % ./star -vvvt test.star
Archivos en la tabla:
Blocks: 6991: 6992 to 6993 6994 to 6995 6996 to 6997 6998 to 6999 7000 to 7001 7002 to 7003 7004 to 7005 7006 to 7007 7008 to 7009 7010 to 7011 7012 to 7013 to 7014
to 7015 7016 to 7017 7018 to 7019 7020 to 7021 7022 to 7023 7024 to 7025 7026 to 7027 7028 to 7029 7030 to 7031 7032 to 7033 7034 to 7035 7036 to 7037 7038 to
7039 7040 to 7041 7042 to 7043 7044 to 7045 7046 to 7047 7048 to 7049 7050 to 7051 7052 to 7053 7054 to 7055 7056 to 7057 7058 to 7059 7060 to 7061 7062 to 706
3 7064 to 7065 7066 to 7067 7068 to 7069 7070
FileHeader: name: 'archival.vot', first: '7071', last: '7071', isDeleted: 0, size: 59, index: 2, blocks: 1.
Blocks: 7071
FileHeader: name: 'otro.txt', first: '7072', last: '7072', isDeleted: 0, size: 47, index: 3, blocks: 1.
Blocks: 7072
FileHeader: name: 'freeBlockHeader', first: 0, last: 6990, isDeleted: 0, size: 407385, index: -1, blocks: 6991.
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33
34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66
67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99
100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127
128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155
156 157 158 159 160 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180 181 182 183
184 185 186 187 188 189 190 191 192 193 194 195 196 197 198 199 200 201 202 203 204 205 206 207 208 209 210 211
212 213 214 215 216 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238
239 240 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255 256 257 258 259 260 261 262 263 264 265 266
267 268 269 270 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287 288 289 290 291 292 293 294
295 296 297 298 299 300 301 302 303 304 305 306 307 308 309 310 311 312 313 314 315 316 317 318 319 320 321 322
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
575 576 577 578 579 580 581 582 583 584 585 586 587 588 589 590 591 592 593 594 595 596 597 598 599 600 601 602
603 604 605 606 607 608 609 610 611 612 613 614 615 616 617 618 619 620 621 622 623 624 625 626 627 628 629 630
631 632 633 634 635 636 637 638 639 640 641 642 643 644 645 646 647 648 649 650 651 652 653 654 655 656 657 658
659 660 661 662 663 664 665 666 667 668 669 670 671 672 673 674 675 676 677 678 679 680 681 682 683 684 685 686
687 688 689 690 691 692 693 694 695 696 697 698 699 700 701 702 703 704 705 706 707 708 709 710 711 712 713 714
715 716 717 718 719 720 721 722 723 724 725 726 727 728 729 730 731 732 733 734 735 736 737 738 739 740 741 742
743 744 745 746 747 748 749 750 751 752 753 754 755 756 757 758 759 760 761 762 763 764 765 766 767 768 769 770
771 772 773 774 775 776 777 778 779 780 781 782 783 784 785 786 787 788 789 790 791 792 793 794 795 796 797 798
799 800 801 802 803 804 805 806 807 808 809 810 811 812 813 814 815 816 817 818 819 820 821 822 823 824 825 826
827 828 829 830 831 832 833 834 835 836 837 838 839 840 841 842 843 844 845 846 847 848 849 850 851 852 853 854
855 856 857 858 859 860 861 862 863 864 865 866 867 868 869 870 871 872 873 874 875 876 877 878 879 880 881 882
883 884 885 886 887 888 889 8
```


2099	2099	2099	2100	2101	2102	2103	2104	2105	2106	2107	2108	2109	2110	2111	2112	2113	2114	2115	2116	2117	2118	2119	2120	2121
122	2122	2123	2124	2125	2126	2127	2128	2129	2130	2131	2132	2133	2134	2135	2136	2137	2138	2139	2140	2141	2142	2143	2144	2145
2146	2147	2148	2149	2150	2151	2152	2153	2154	2155	2156	2157	2158	2159	2160	2161	2162	2163	2164	2165	2166	2167	2168	2169	2170
2171	2172	2173	2174	2175	2176	2177	2178	2179	2180	2181	2182	2183	2184	2185	2186	2187	2188	2189	2190	2191	2192	2193	2194	2195
2196	2197	2198	2199	2200	2201	2202	2203	2204	2205	2206	2207	2208	2209	2210	2211	2212	2213	2214	2215	2216	2217	2218	2219	2220
2221	2222	2223	2224	2225	2226	2227	2228	2229	2230	2231	2232	2233	2234	2235	2236	2237	2238	2239	2240	2241	2242	2243	2244	2245
2246	2247	2248	2249	2250	2251	2252	2253	2254	2255	2256	2257	2258	2259	2260	2261	2262	2263	2264	2265	2266	2267	2268	2269	
2270	2271	2272	2273	2274	2275	2276	2277	2278	2279	2280	2281	2282	2283	2284	2285	2286	2287	2288	2289	2290	2291	2292	2293	
2294	2295	2296	2297	2298	2299	2300	2301	2302	2303	2304	2305	2306	2307	2308	2309	2310	2311	2312	2313	2314	2315	2316	2317	
2318	2319	2320	2321	2322	2323	2324	2325	2326	2327	2328	2329	2330	2331	2332	2333	2334	2335	2336	2337	2338	2339	2340	2341	
2342	2343	2344	2345	2346	2347	2348	2349	2350	2351	2352	2353	2354	2355	2356	2357	2358	2359	2360	2361	2362	2363	2364	2365	
2366	2367	2368	2369	2370	2371	2372	2373	2374	2375	2376	2377	2378	2379	2380	2381	2382	2383	2384	2385	2386	2387	2388	2389	
2390	2391	2392	2393	2394	2395	2396	2397	2398	2399	2400	2401	2402	2403	2404	2405	2406	2407	2408	2409	2410	2411	2412	2413	
2414	2415	2416	2417	2418	2419	2420	2421	2422	2423	2424	2425	2426	2427	2428	2429	2430	2431	2432	2433	2434	2435	2436	2437	
2438	2439	2440	2441	2442	2443	2444	2445	2446	2447	2448	2449	2450	2451	2452	2453	2454	2455	2456	2457	2458	2459	2460	2461	
2462	2463	2464	2465	2466	2467	2468	2469	2470	2471	2472	2473	2474	2475	2476	2477	2478	2479	2480	2481	2482	2483	2484	2485	
2486	2487	2488	2489	2490	2491	2492	2493	2494	2495	2496	2497	2498	2499	2500	2501	2502	2503	2504	2505	2506	2507	2508	2509	
2510	2511	2512	2513	2514	2515	2516	2517	2518	2519	2520	2521	2522	2523	2524	2525	2526	2527	2528	2529	2530	2531	2532	2533	
2534	2535	2536	2537	2538	2539	2540	2541	2542</																

1	5013	5014	5015	5016	5017	5018		5020	5021	5022	5023	5024	5025	5026	5027	5028	5029	5030	5031	5032	5033	5034	5035	5036	5037
5038	5039	5040	5041	5042	5043	5044	5045	5046	5047	5048	5049	5050	5051	5052	5053	5054	5055	5056	5057	5058	5059	5060	5061	5062	5063
5064	5065	5066	5067	5068	5069	5070	5071	5072	5073	5074	5075	5076	5077	5078	5079	5080	5081	5082	5083	5084	5085	5086	5087	5088	
5089	5090	5091	5092	5093	5094	5095	5096	5097	5098	5099	5100	5101	5102	5103	5104	5105	5106	5107	5108	5109	5110	5111	5112	5113	
5114	5115	5116	5117	5118	5119	5120	5121	5122	5123	5124	5125	5126	5127	5128	5129	5130	5131	5132	5133	5134	5135	5136	5137	5138	
5139	5140	5141	5142	5143	5144	5145	5146	5147	5148	5149	5150	5151	5152	5153	5154	5155	5156	5157	5158	5159	5160	5161	5162	5163	
5164	5165	5166	5167	5168	5169	5170	5171	5172	5173	5174	5175	5176	5177	5178	5179	5180	5181	5182	5183	5184	5185	5186	5187	5188	
5189	5190	5191	5192	5193	5194	5195	5196	5197	5198	5199	5200	5201	5202	5203	5204	5205	5206	5207	5208	5209	5210	5211	5212	5213	
5214	5215	5216	5217	5218	5219	5220	5221	5222	5223	5224	5225	5226	5227	5228	5229	5230	5231	5232	5233	5234	5235	5236	5237	5238	
5239	5240	5241	5242	5243	5244	5245	5246	5247	5248	5249	5250	5251	5252	5253	5254	5255	5256	5257	5258	5259	5260	5261	5262	5263	
5264	5265	5266	5267	5268	5269	5270	5271	5272	5273	5274	5275	5276	5277	5278	5279	5280	5281	5282	5283	5284	5285	5286	5287	5288	
5289	5290	5291	5292	5293	5294	5295	5296	5297	5298	5299	5300	5301	5302	5303	5304	5305	5306	5307	5308	5309	5310	5311	5312	5313	
5314	5315	5316	5317	5318	5319	5320	5321	5322	5323	5324	5325	5326	5327	5328	5329	5330	5331	5332	5333	5334	5335	5336	5337	5338	
5339	5340	5341	5342	5343	5344	5345	5346	5347	5348	5349	5350	5351	5352	5353	5354	5355	5356	5357	5358	5359	5360	5361	5362	5363	
5364	5365	5366	5367	5368	5369	5370	5371	5372	5373	5374	5375	5376	5377	5378	5379	5380	5381	5382	5383	5384	5385	5386	5387	5388	
5389	5390	5391	5392	5393	5394	5395	5396	5397	5398	5399	5400	5401	5402	5403	5404	5405	5406	5407	5408	5409	5410	5411	5412	5413	
5414	5415	5416	5417	5418	5419	5420	5421	5422	5423	5424	5425	5426	5427	5428	5429	5430	5431	5432	5433	5434	5435	5436	5437	5438	
5439	5440	5441	5442	5443	5444	5445	5446	5447	5448	5449	5450	5451	5452	5453	5454	5455	5456	5457	5458	5459					

```
--> 6508 -> 6509 -> 6510 -> 6511 -> 6512 -> 6513 -> 6514 -> 6515 -> 6516 -> 6517 -> 6518 -> 6519 -> 6520 -> 6521 -> 6522 -> 6523 -> 6524 -> 6525 -> 6526 -> 6527 -> 6528 -> 6529 -> 6530 -> 6531 -> 6
532 -> 6533 -> 6534 -> 6535 -> 6536 -> 6537 -> 6538 -> 6539 -> 6540 -> 6541 -> 6542 -> 6543 -> 6544 -> 6545 -> 6546 -> 6547 -> 6548 -> 6549 -> 6550 -> 6551 -> 6552 -> 6553 -> 6554 -> 6555 -> 6556
-> 6557 -> 6558 -> 6559 -> 6560 -> 6561 -> 6562 -> 6563 -> 6564 -> 6565 -> 6566 -> 6567 -> 6568 -> 6569 -> 6570 -> 6571 -> 6572 -> 6573 -> 6574 -> 6575 -> 6576 -> 6577 -> 6578 -> 6579 -> 6580 -> 6
581 -> 6582 -> 6583 -> 6584 -> 6585 -> 6586 -> 6587 -> 6588 -> 6589 -> 6590 -> 6591 -> 6592 -> 6593 -> 6594 -> 6595 -> 6596 -> 6597 -> 6598 -> 6599 -> 6600 -> 6601 -> 6602 -> 6603 -> 6604 -> 6605
-> 6606 -> 6607 -> 6608 -> 6609 -> 6610 -> 6611 -> 6612 -> 6613 -> 6614 -> 6615 -> 6616 -> 6617 -> 6618 -> 6619 -> 6620 -> 6621 -> 6622 -> 6623 -> 6624 -> 6625 -> 6626 -> 6627 -> 6628 -> 6629 -> 6
630 -> 6631 -> 6632 -> 6633 -> 6634 -> 6635 -> 6636 -> 6637 -> 6638 -> 6639 -> 6640 -> 6641 -> 6642 -> 6643 -> 6644 -> 6645 -> 6646 -> 6647 -> 6648 -> 6649 -> 6650 -> 6651 -> 6652 -> 6653 -> 6654
-> 6655 -> 6656 -> 6657 -> 6658 -> 6659 -> 6660 -> 6661 -> 6662 -> 6663 -> 6664 -> 6665 -> 6666 -> 6667 -> 6668 -> 6669 -> 6670 -> 6671 -> 6672 -> 6673 -> 6674 -> 6675 -> 6676 -> 6677 -> 6678 -> 6
679 -> 6680 -> 6681 -> 6682 -> 6683 -> 6684 -> 6685 -> 6686 -> 6687 -> 6688 -> 6689 -> 6690 -> 6691 -> 6692 -> 6693 -> 6694 -> 6695 -> 6696 -> 6697 -> 6698 -> 6699 -> 6700 -> 6701 -> 6702 -> 6703
-> 6704 -> 6705 -> 6706 -> 6707 -> 6708 -> 6709 -> 6710 -> 6711 -> 6712 -> 6713 -> 6714 -> 6715 -> 6716 -> 6717 -> 6718 -> 6719 -> 6720 -> 6721 -> 6722 -> 6723 -> 6724 -> 6725 -> 6726 -> 6727 -> 6
728 -> 6729 -> 6730 -> 6731 -> 6732 -> 6733 -> 6734 -> 6735 -> 6736 -> 6737 -> 6738 -> 6739 -> 6740 -> 6741 -> 6742 -> 6743 -> 6744 -> 6745 -> 6746 -> 6747 -> 6748 -> 6749 -> 6750 -> 6751 -> 6752
-> 6753 -> 6754 -> 6755 -> 6756 -> 6757 -> 6758 -> 6759 -> 6760 -> 6761 -> 6762 -> 6763 -> 6764 -> 6765 -> 6766 -> 6767 -> 6768 -> 6769 -> 6770 -> 6771 -> 6772 -> 6773 -> 6774 -> 6775 -> 6776 -> 6
777 -> 6778 -> 6779 -> 6780 -> 6781 -> 6782 -> 6783 -> 6784 -> 6785 -> 6786 -> 6787 -> 6788 -> 6789 -> 6790 -> 6791 -> 6792 -> 6793 -> 6794 -> 6795 -> 6796 -> 6797 -> 6798 -> 6799 -> 6800 -> 6801
-> 6802 -> 6803 -> 6804 -> 6805 -> 6806 -> 6807 -> 6808 -> 6809 -> 6810 -> 6811 -> 6812 -> 6813 -> 6814 -> 6815 -> 6816 -> 6817 -> 6818 -> 6819 -> 6820 -> 6821 -> 6822 -> 6823 -> 6824 -> 6825 -> 6
826 -> 6827 -> 6828 -> 6829 -> 6830 -> 6831 -> 6832 -> 6833 -> 6834 -> 6835 -> 6836 -> 6837 -> 6838 -> 6839 -> 6840 -> 6841 -> 6842 -> 6843 -> 6844 -> 6845 -> 6846 -> 6847 -> 6848 -> 6849 -> 6850
-> 6851 -> 6852 -> 6853 -> 6854 -> 6855 -> 6856 -> 6857 -> 6858 -> 6859 -> 6860 -> 6861 -> 6862 -> 6863 -> 6864 -> 6865 -> 6866 -> 6867 -> 6868 -> 6869 -> 6870 -> 6871 -> 6872 -> 6873 -> 6874 -> 6
875 -> 6876 -> 6877 -> 6878 -> 6879 -> 6880 -> 6881 -> 6882 -> 6883 -> 6884 -> 6885 -> 6886 -> 6887 -> 6888 -> 6889 -> 6890 -> 6891 -> 6892 -> 6893 -> 6894 -> 6895 -> 6896 -> 6897 -> 6898 -> 6899
-> 6900 -> 6901 -> 6902 -> 6903 -> 6904 -> 6905 -> 6906 -> 6907 -> 6908 -> 6909 -> 6910 -> 6911 -> 6912 -> 6913 -> 6914 -> 6915 -> 6916 -> 6917 -> 6918 -> 6919 -> 6920 -> 6921 -> 6922 -> 6923 -> 6
924 -> 6925 -> 6926 -> 6927 -> 6928 -> 6929 -> 6930 -> 6931 -> 6932 -> 6933 -> 6934 -> 6935 -> 6936 -> 6937 -> 6938 -> 6939 -> 6940 -> 6941 -> 6942 -> 6943 -> 6944 -> 6945 -> 6946 -> 6947 -> 6948
-> 6949 -> 6950 -> 6951 -> 6952 -> 6953 -> 6954 -> 6955 -> 6956 -> 6957 -> 6958 -> 6959 -> 6960 -> 6961 -> 6962 -> 6963 -> 6964 -> 6965 -> 6966 -> 6967 -> 6968 -> 6969 -> 6970 -> 6971 -> 6972 -> 6
973 -> 6974 -> 6975 -> 6976 -> 6977 -> 6978 -> 6979 -> 6980 -> 6981 -> 6982 -> 6983 -> 6984 -> 6985 -> 6986 -> 6987 -> 6988 -> 6989 -> 6990
```

4. Pack de archivos:

Contenido actual de la tabla de archivos

```
● (base) franvq09@MBPdeFrancisco S-tar % ./star -vvvt test.star
argc: 3
Archivos en la tabla:
FileHeader: name: 'archivo1.txt', first: 0, last: 0, isDeleted: 0, size: 32, index: 0, blocks: 1.
Blocks : 0
FileHeader: name: 'otro.txt', first: 2, last: 2, isDeleted: 0, size: 44, index: 2, blocks: 1.
Blocks : 2
FileHeader: name: 'freeBlocksHeader', first: 1, last: 1, isDeleted: 0, size: 45, index: -1, blocks: 1.
Blocks : 1
```

Ejecución del comando pack

```
● (base) franvq09@MBPdeFrancisco S-tar % ./star -vvvp test.star
argc: 3
```

```
● (base) franvq09@MBPdeFrancisco S-tar % ./star -vvvt test.star
argc: 3
Archivos en la tabla:
FileHeader: name: 'archivo1.txt', first: 0, last: 0, isDeleted: 0, size: 32, index: 0, blocks: 1.
Blocks : 0
FileHeader: name: 'otro.txt', first: 1, last: 1, isDeleted: 0, size: 44, index: 2, blocks: 1.
Blocks : 1
FileHeader: name: 'freeBlocksHeader', first: -1, last: -1, isDeleted: 0, size: 0, index: -1, blocks: 0.
Blocks : no blocks
```

Como se puede observar, los bloques libres resultantes del delete, se vuelven a acomodar con los archivos actuales.

4.4 Pack de archivos con diferentes extensiones:

Contenido actual de la table de archivos

```
● (base) franvq09@MBPdeFrancisco S-tar % ./star -vvvt test.star
argc: 3
Archivos en la tabla:
FileHeader: name: 'archivo1.txt', first: 0, last: 0, isDeleted: 0, size: 32, index: 0, blocks: 1.
Blocks : 0
FileHeader: name: 'archivo2.txt', first: 1, last: 1, isDeleted: 0, size: 45, index: 1, blocks: 1.
Blocks : 1
FileHeader: name: 'otro.txt', first: 2, last: 2, isDeleted: 0, size: 44, index: 2, blocks: 1.
Blocks : 2
FileHeader: name: 'Hola.pdf', first: 3, last: 281, isDeleted: 0, size: 17803, index: 3, blocks: 279.
Blocks : 3 -> 4 -> 5 -> 6 -> 7 -> 8 -> 9 -> 10 -> 11 -> 12 -> 13 -> 14 -> 15 -> 16 -> 17 -> 18 -> 19 -> 20 -> 21 -> 22 -> 23 -> 24 -> 25 -> 26 -> 27 -> 28 -> 29 -> 30 -> 31 -> 32
-> 33 -> 34 -> 35 -> 36 -> 37 -> 38 -> 39 -> 40 -> 41 -> 42 -> 43 -> 44 -> 45 -> 46 -> 47 -> 48 -> 49 -> 50 -> 51 -> 52 -> 53 -> 54 -> 55 -> 56 -> 57 -> 58 -> 59 -> 60 -> 61 -> 62
-> 63 -> 64 -> 65 -> 66 -> 67 -> 68 -> 69 -> 70 -> 71 -> 72 -> 73 -> 74 -> 75 -> 76 -> 77 -> 78 -> 79 -> 80 -> 81 -> 82 -> 83 -> 84 -> 85 -> 86 -> 87 -> 88 -> 89 -> 90 -> 91 -> 9
2 -> 93 -> 94 -> 95 -> 96 -> 97 -> 98 -> 99 -> 100 -> 101 -> 102 -> 103 -> 104 -> 105 -> 106 -> 107 -> 108 -> 109 -> 110 -> 111 -> 112 -> 113 -> 114 -> 115 -> 116 -> 117 -> 118 ->
119 -> 120 -> 121 -> 122 -> 123 -> 124 -> 125 -> 126 -> 127 -> 128 -> 129 -> 130 -> 131 -> 132 -> 133 -> 134 -> 135 -> 136 -> 137 -> 138 -> 139 -> 140 -> 141 -> 142 -> 143 -> 144
-> 145 -> 146 -> 147 -> 148 -> 149 -> 150 -> 151 -> 152 -> 153 -> 154 -> 155 -> 156 -> 157 -> 158 -> 159 -> 160 -> 161 -> 162 -> 163 -> 164 -> 165 -> 166 -> 167 -> 168 -> 169 ->
170 -> 171 -> 172 -> 173 -> 174 -> 175 -> 176 -> 177 -> 178 -> 179 -> 180 -> 181 -> 182 -> 183 -> 184 -> 185 -> 186 -> 187 -> 188 -> 189 -> 190 -> 191 -> 192 -> 193 -> 194 -> 195
-> 196 -> 197 -> 198 -> 199 -> 200 -> 201 -> 202 -> 203 -> 204 -> 205 -> 206 -> 207 -> 208 -> 209 -> 210 -> 211 -> 212 -> 213 -> 214 -> 215 -> 216 -> 217 -> 218 -> 219 -> 220 -> 2
21 -> 222 -> 223 -> 224 -> 225 -> 226 -> 227 -> 228 -> 229 -> 230 -> 231 -> 232 -> 233 -> 234 -> 235 -> 236 -> 237 -> 238 -> 239 -> 240 -> 241 -> 242 -> 243 -> 244 -> 245 -> 246 ->
247 -> 248 -> 249 -> 250 -> 251 -> 252 -> 253 -> 254 -> 255 -> 256 -> 257 -> 258 -> 259 -> 260 -> 261 -> 262 -> 263 -> 264 -> 265 -> 266 -> 267 -> 268 -> 269 -> 270 -> 271 -> 27
2 -> 273 -> 274 -> 275 -> 276 -> 277 -> 278 -> 279 -> 280 -> 281
FileHeader: name: 'freeBlocksHeader', first: -1, last: -1, isDeleted: 0, size: 0, index: -1, blocks: 0.
Blocks : no blocks
```

Ejecución del comando pack:


```
● (base) franvq09@MBPdeFrancisco S-tar % ./star -vvvp test.star  
argc: 3
```

```
● (base) franvq09@MBPdeFrancisco S-tar % ./star -vvvt test.star  
argc: 3  
Archivos en la tabla:  
FileHeader: name: 'archivo1.txt', first: 0, last: 0, isDeleted: 0, size: 32, index: 0, blocks: 1.  
Blocks : 0  
FileHeader: name: 'archivo2.txt', first: 1, last: 1, isDeleted: 0, size: 45, index: 1, blocks: 1.  
Blocks : 1  
FileHeader: name: 'otro.txt', first: 2, last: 2, isDeleted: 0, size: 44, index: 2, blocks: 1.  
Blocks : 2  
FileHeader: name: 'Hola.pdf', first: 3, last: 281, isDeleted: 0, size: 17803, index: 3, blocks: 279.  
Blocks : 3 -> 4 -> 5 -> 6 -> 7 -> 8 -> 9 -> 10 -> 11 -> 12 -> 13 -> 14 -> 15 -> 16 -> 17 -> 18 -> 19 -> 20 -> 21 -> 22 -> 23 -> 24 -> 25 -> 26 -> 27 -> 28 -> 29 -> 30 -> 31 -> 32  
-> 33 -> 34 -> 35 -> 36 -> 37 -> 38 -> 39 -> 40 -> 41 -> 42 -> 43 -> 44 -> 45 -> 46 -> 47 -> 48 -> 49 -> 50 -> 51 -> 52 -> 53 -> 54 -> 55 -> 56 -> 57 -> 58 -> 59 -> 60 -> 61 -> 62  
-> 63 -> 64 -> 65 -> 66 -> 67 -> 68 -> 69 -> 70 -> 71 -> 72 -> 73 -> 74 -> 75 -> 76 -> 77 -> 78 -> 79 -> 80 -> 81 -> 82 -> 83 -> 84 -> 85 -> 86 -> 87 -> 88 -> 89 -> 90 -> 91 -> 9  
2 -> 93 -> 94 -> 95 -> 96 -> 97 -> 98 -> 99 -> 100 -> 101 -> 102 -> 103 -> 104 -> 105 -> 106 -> 107 -> 108 -> 109 -> 110 -> 111 -> 112 -> 113 -> 114 -> 115 -> 116 -> 117 -> 118 ->  
119 -> 120 -> 121 -> 122 -> 123 -> 124 -> 125 -> 126 -> 127 -> 128 -> 129 -> 130 -> 131 -> 132 -> 133 -> 134 -> 135 -> 136 -> 137 -> 138 -> 139 -> 140 -> 141 -> 142 -> 143 -> 144  
-> 145 -> 146 -> 147 -> 148 -> 149 -> 150 -> 151 -> 152 -> 153 -> 154 -> 155 -> 156 -> 157 -> 158 -> 159 -> 160 -> 161 -> 162 -> 163 -> 164 -> 165 -> 166 -> 167 -> 168 -> 169 ->  
170 -> 171 -> 172 -> 173 -> 174 -> 175 -> 176 -> 177 -> 178 -> 179 -> 180 -> 181 -> 182 -> 183 -> 184 -> 185 -> 186 -> 187 -> 188 -> 189 -> 190 -> 191 -> 192 -> 193 -> 194 -> 195  
-> 196 -> 197 -> 198 -> 199 -> 200 -> 201 -> 202 -> 203 -> 204 -> 205 -> 206 -> 207 -> 208 -> 209 -> 210 -> 211 -> 212 -> 213 -> 214 -> 215 -> 216 -> 217 -> 218 -> 219 -> 220 -> 2  
21 -> 222 -> 223 -> 224 -> 225 -> 226 -> 227 -> 228 -> 229 -> 230 -> 231 -> 232 -> 233 -> 234 -> 235 -> 236 -> 237 -> 238 -> 239 -> 240 -> 241 -> 242 -> 243 -> 244 -> 245 -> 246 ->  
2 -> 247 -> 248 -> 249 -> 250 -> 251 -> 252 -> 253 -> 254 -> 255 -> 256 -> 257 -> 258 -> 259 -> 260 -> 261 -> 262 -> 263 -> 264 -> 265 -> 266 -> 267 -> 268 -> 269 -> 270 -> 271 -> 27  
2 -> 273 -> 274 -> 275 -> 276 -> 277 -> 278 -> 279 -> 280 -> 281  
FileHeader: name: 'freeBlocksHeader', first: -1, last: -1, isDeleted: 0, size: 0, index: -1, blocks: 0.  
Blocks : no blocks
```

Conclusiones

- El objetivo principal del proyecto se cumplió al poder tener una aplicación que gestione bloques, cree, elimine, actualice, agregue y desfragmente.
- El uso de los comandos se puede mejorar al permitir más mezcla entre ellos, además de brindar validaciones más estrictas y una guía de uso.
- El manejo de la desfragmentación mueve los bloques al inicio del archivo permitiendo eliminar la cola de bloques libres. Este proceso tiene cabida para la optimización de un algoritmo más eficiente.
- Durante el desarrollo se encontraron dificultades al plantear el problema pero se logró un manejo correcto de los archivos en disco, sin tener que cargar toda la información a memoria.