Operating Systems — Coursework 2

Axel Goetz

November 29, 2016

There were several different inconsistencies that this projects detects and fixes:

1. Finds all unreferenced clusters and logs them in the following format: `Unreferenced:<cluster_number>`.
2. Checks the unreferenced clusters for lost files. Then logs them as follows `Lost File:<start_cluster> <size_in_clusters>` and creates a file in the root directory named `FOUND<file_number>.dat`.
3. Locates all of the files whose length in the FAT is inconsistent with the length in the directory entry. Then it logs all of them in the format `<filename> <fat_length> <directory_length>`.
4. For all the previously located files, it frees any clusters beyond the end of the file.

To achieve these goals, the coursework makes use of a similar infrastructure provided in both `dos_ls.c` and `dos_cp.c`. Below you can see how each individual inconsistency was detected and fixed:

1. The program creates a boolean array called `referenced_clusters` with all values initialised to `false`. Next, it goes through the directory tree and every file in the tree and marks the value `referenced_clusters[cluster_number]` as `true`. After traversing the entire tree recursively, you then know which clusters have not been referenced by any file. Next, the program iterates through the `referenced_cluster` and checks if the value is set to `false` and if the cluster is not free. If that is the case, it prints the cluster to `stdout`.
2. Next, uses a very similar technique to locate the lost files. Again, the program iterates through the `referenced_clusters` array. If the value is set to `false` and the cluster is not free, the program calculates the size of the file by visiting all of the FAT values, starting from the cluster that was not referenced. Finally it creates a new file in the root directory and sets the appropriate name, extension, size and start cluster. This approach assumes that a file always starts with the lowest cluster. For instance a file might consist of the clusters $3 \rightarrow 6 \rightarrow 4 \rightarrow 9 \rightarrow 10$ but not $6 \rightarrow 3 \rightarrow 4 \rightarrow 9 \rightarrow 10$ since $6 > \min(clusters)$.
3. To check if all file lengths are consistent, the program again iterates through all directories recursively and checks the file length. This definitely checks every file because we previously creates a new file in the root directory for every unreferenced file. If the file length is inconsistent with the one in the FAT, it then frees up the memory as described in the next step.
4. To free all of blocks beyond the end of that file, the program simply frees all of the clusters beyond the end and then sets the appropriate cluster as the end of file (EOF).