This document is a quick introduction to the layout of the “ROM Disk” supported in 68K/OS

Most programs and procedure files have a 32 byte header, but this isn’t the case with every file (though these are exceptions with smaller headers, but they work fine as long as the basics are there)

00 00 00 20

The first 4 bytes are the relative offset to the entry point in the PROGram
This is usually $20, but not always

00 04

The next 2 bytes are how much memory needs to be allocated (in 1Kb lumps)
Setting this to 00 00 flags the file as a PROCedure and means it can’t be run as a program

00 06

The next 2 bytes are the filename length

MYPROG

The ASCII Program name
Not necessarily the Filename, though it could be
You could use myprog.prog if preferred, but unnecessary

00

If the filename is an odd number of characters then the next byte is 00
This is to keep everything on a Word Boundary

00 x ?

The remaining bytes, to pack it to 32, can just be 00

The code & data etc. follows the header of course

ROM Disk

The ROM Disk is similar to the PROG/PROC files, but headers are NOT 32 bytes

ROM:

The first 4 bytes are the text “ROM:”

00 01

The next 2 bytes seem, currently, to serve no purpose. “00 01” is what is used in the main ROM for ADAM

Each file directory entry follows on from the next and are structured as follows

00 00 0F 20

The first 4 bytes are the relative offset to the entry point of the PROGram in ROM

00 01

The next 2 bytes are how much memory needs to be allocated (in 1Kb lumps)
Setting this to 00 00 flags the file as a PROCedure and means it can’t be run
This information can be obtained from the header of the files being loaded to ROM

00 06

The next 2 bytes are the filename length

MYPROG

The ASCII Program name, and will be the name in the directory listing
You COULD include .PROG but it’s pointless as you’ll have to type it each time
You do need to include PROC if it’s a procedure file

00

If the filename is an odd number of characters then add a 00 byte

> DO NOT PACK OUT ANY FURTHER
> THE NEXT DIRECTORY ENTRY SHOULD START HERE

00 00 00 00

After the final directory entry 4 x $00 bytes are required to indicate end of directory

The rest of the ROM is then filled with the PROG/PROC files (without their headers, but you could leave them in for clarity I suppose if space isn’t a problem)

Remember the relative entry point has to be calculated from the relative entry point in the original file header and the position of the code in the ROM image