$[CHAR]$
$STATE ::= READ \mid WRITE$
$RESPONSE ::= OK \mid FILE\_WRONG\_MODE$

$\mid \quad EOF : CHAR$

$STRING == seq\,(CHAR \setminus \{EOF\})$

File

$content : STRING$
$state : STATE$
$position : \mathbb{N}$

$WriteFail == [\Xi File;\ resp! : RESPONSE \setminus \{OK\} \mid state = READ \Rightarrow resp! = FILE\_WRONG\_MODE]$
$ReadFail == [\Xi File;\ resp! : RESPONSE \setminus \{OK\} \mid state = WRITE \Rightarrow resp! = FILE\_WRONG\_MODE]$

FileInit

$File'$

$content' = \langle\rangle$
$state' = WRITE$

Overwrite

$\Delta File$
$in? : STRING$

$state = WRITE$
$content' = in?$
$state' = state$

Note "[Overwrite; resp!: {OK}]" could be replaced with a more general 'OK' Schema, and ANDED with the rest of the Total schema. However I think the way below potentially illustrates what's going on better.

$OverwriteTotal == [Overwrite;\ resp! : \{OK\}] \lor WriteFail$

Append

$\Delta File$
$in? : CHAR \setminus \{EOF\}$

$state = WRITE$
$content' = content \frown \langle in? \rangle$
$state' = state$

$AppendTotal == [Append;\ resp! : \{OK\}] \lor WriteFail$

If a user tries to reopen an open file it will not reset the read position.

Open

$\Delta File$

$state = WRITE \Rightarrow state' = READ \land position' = 0$
$state = READ \Rightarrow state' = state \land position' = position$
$content' = content$

$OpenTotal == [Open;\ resp! : \{OK\}] \lor WriteFail$

ReadChar

$\Delta File$
$out! : CHAR$

$state = READ$
$content' = content$
$state' = state$
$position \in dom\ content \Rightarrow out! = content\ position \land position' = position + 1$
$position \notin dom\ content \Rightarrow out! = EOF \land position' \notin dom\ content$

$ReadCharTotal == [ReadChar;\ resp! : \{OK\}] \lor ReadFail$

Close

$\Delta File$

$state = READ$
$state' = WRITE$
$content' = content$

$CloseTotal == [Close;\ resp! : \{OK\}] \lor ReadFail$

$[LOC]$
$FileSystem == [files : LOC \pfun File]$

A filesystem is a partial function between Locations (inodes?) and Files, as defined above.
Currently it's got unlimited size.

Promote

$\Delta FileSystem$
$\Delta File$
$loc? : LOC$

$loc? \in dom\ files$
$files\ loc? = \theta\ File$
$files' = files \oplus \{loc? \mapsto \theta\ File'\}$

To create a new file, we need to Initialise some anonymous file with a location (name).
It must not already exist. We add it to the files map.

$NewFilePrelim == [\Delta FileSystem;\ FileInit;\ loc? : LOC \mid loc? \notin dom\ files;\ files' = files \oplus \{loc? \mapsto \theta\ File'\}]$
$NewFile == \exists File' \bullet NewFilePrelim$

We can delete a file, if it exists. We use domain substraction so we only need to specify the name, and not the name (loc?) and not loc? and the 'File' it maps to.

$$DeleteFile == [\Delta FileSystem;\ loc? : LOC \mid loc? \in dom\, files;\ files' = \{loc?\} \lhd files]$$
$$OverwriteFile == \exists\, \Delta File \bullet Promote \land OverwriteTotal$$
$$AppendFile == \exists\, \Delta File \bullet Promote \land AppendTotal$$
$$OpenFile == \exists\, \Delta File \bullet Promote \land OpenTotal$$
$$ReadCharFile == \exists\, \Delta File \bullet Promote \land ReadCharTotal$$
$$CloseFile == \exists\, \Delta File \bullet Promote \land CloseTotal$$