MODULE *storagecleanernaive*

Cleaner States

```
$CleanerStartGetBlobKeys(c) \triangleq$
    LET $current \triangleq cleanerStates[c]$ IN
      Starts only from waiting
      $\land\ current.state = $ "waiting"
      $\land\ cleanerStates' = [$
          $cleanerStates$ EXCEPT
              $![c].state = $ "got_blob_keys",
               All keys that are set in blockstore
              $![c].blobKeys = \{k \in UUIDS : blobStoreState[k] \neq $ "UNSET"$\}$
          $]$
      $\land$ UNCHANGED $\langle serverStates, databaseState, blobStoreState, operations\rangle$

$CleanerGetUnusedKeys(c) \triangleq$
    LET $current \triangleq cleanerStates[c]$ IN
      From blob keys, get unused keys from database
      $\land\ current.state = $ "got_blob_keys"
      $\land\ cleanerStates' = [$
          $cleanerStates$ EXCEPT
              $![c].state = $ "got_unused_keys",
              $![c].unusedBlobKeys =$
                  $\{k \in current.blobKeys :$ Keys in blob keys
                      $\forall u \in USERIDS :$ That are not in the database
                          $databaseState[u].imageId \neq k\}$
          $]$
      $\land$ UNCHANGED $\langle serverStates, databaseState, blobStoreState, operations\rangle$

$CleanerDeletingKeys(c) \triangleq$
    LET $current \triangleq cleanerStates[c]$ IN
      When we have unused keys, keep deleting
      $\land\ current.state \in \{$"got_unused_keys", "deleting_keys"$\}$
      $\land\ Cardinality(current.unusedBlobKeys) \neq 0$
      $\land\ \exists k \in current.unusedBlobKeys :$ Pick a key to delete
          $\land\ blobStoreState' = [blobStoreState$ EXCEPT $![k] = $ "UNSET"$]$
          $\land\ cleanerStates' = [$
              $cleanerStates$ EXCEPT
                   Remove the key from set
                  $![c].unusedBlobKeys = current.unusedBlobKeys \setminus \{k\}$
              $]$
      $\land$ UNCHANGED $\langle serverStates, databaseState, operations\rangle$

$CleanerFinished(c) \triangleq$
    LET $current \triangleq cleanerStates[c]$ IN
      ∧ $current.state$ = "deleting_keys"
      When we have no more unused keys to delete, finish
      ∧ $Cardinality(current.unusedBlobKeys) = 0$
      ∧ $cleanerStates' =$ [
          $cleanerStates$ EXCEPT
              ![$c$].$state$ = "waiting",
              ![$c$].$blobKeys$ = {},
              ![$c$].$unusedBlobKeys$ = {}
          ]
      ∧ UNCHANGED ⟨$serverStates$, $databaseState$, $blobStoreState$, $operations$⟩

  $CleanerFail(c) \triangleq$
      LET $current \triangleq cleanerStates[c]$ IN
      Cleaner can fail from any active state
      ∧ $current.state \in$ {"got_blob_keys", "got_unused_keys", "deleting_keys"}
      Failure represented by cleaner losing state. Any partial operations stay partially finished.
      ∧ $cleanerStates' =$ [
          $cleanerStates$ EXCEPT
              ![$c$].$state$ = "waiting",
              ![$c$].$blobKeys$ = {},
              ![$c$].$unusedBlobKeys$ = {}
          ]
      ∧ UNCHANGED ⟨$serverStates$, $databaseState$, $blobStoreState$, $operations$⟩
```

Specification / *Next*

```
$Next \triangleq$
    For every step, we either trigger a server or cleaner to take a step
    ∨ $\exists s \in SERVERS$ :
          ∨ $ServerStartWrite(s)$
          ∨ $ServerWriteBlob(s)$
          ∨ $ServerWriteMetadataAndReturn(s)$
          ∨ $ServerFailWrite(s)$
          ∨ $ServerStartRead(s)$
          ∨ $ServerReadMetadata(s)$
          ∨ $ServerReadMetadataAndReturnEmpty(s)$
          ∨ $ServerReadBlobAndReturn(s)$
    ∨ $\exists c \in CLEANERS$ :  All the steps a cleaner can take
          ∨ $CleanerStartGetBlobKeys(c)$
          ∨ $CleanerGetUnusedKeys(c)$
          ∨ $CleanerDeletingKeys(c)$
          ∨ $CleanerFinished(c)$
          ∨ $CleanerFail(c)$


$Spec \triangleq Init \land \Box[Next]_{vars}$
```

Invariants

```
$NoOrphanFiles \triangleq$
      There does not exist a key
    $\neg \exists k \in UUIDS :$
        That is in the block store
        $\land blobStoreState[k] \neq$ "UNSET"
        And not in the database
        $\land \forall u \in USERIDS :$
            $databaseState[u].imageId \neq k$
```