Dramatically Reduce the Cost of Sequential File Accesses in CICS

Stephen Reid
StephenPReid.com

Friday March 6, 2015
Session Number 16581

#SHAREorg

SHARE is an independent volunteer-run information technology association that provides **education**, professional **networking** and industry **influence**.

SHARE in Seattle 2015

Agenda

- Background
- Requirements
- Solution
- Implementation
- Refinements and Extensions
- Making it all Threadsafe
- 64 Bit - A Whole New World !
- Questions

Background

- It all started with 9/11
- FBI mandate to screen all financial transactions
- 15 million SWIFT transactions per day
- Typically ~50 fields of ~100 characters, per transaction
- Need to check each field against every suspect name
- Fuzzy match on 20,000 names initially – and growing!
- Benchmark proved impossible with normal access methods
- Asked to design/develop a **super efficient** data access
- >500% faster than required access speed
- Fuzzy match algorithm a story in itself – for another time . . .

Requirements

- Read the “Next Record” with minimum machine instructions
- Allow multiple (unlimited) simultaneous Read accesses
- Avoid “Below-the-Line” storage overheads
- Avoid Open/Close overheads (x15 million/day)
- (Allow flexibility in Record Length)

Possible Extra Requirements (not for FBI)

- The following functions introduce Threadsafe issues: (colour-coded blue in subsequent slides)
- Support real-time Updates, Additions and Deletions (ESDS)
- Ensure any changes are controlled and secure
- Ensure data is always Current
- Prevent “Double Updates”
- Support variable-length records

Solution

- Main Memory ! (20,000 X 80 bytes = only 1.6M)
- Allocate a Linked List of Record “Cells” Above the 16M Line
- Store Control Information in a CICS Table (28 byte CSECT)
- Make Control Table “Resident”, so never freed
- Resident means it occupies only 32 bytes, not 4K
- Preload the file during PLTPI
- Access Method only involved **once** at CICS Startup
- Subsequent “READ” of each Record just moves its address
- 3 Machine Instructions instead of at least several hundred
- If CICS dies, PLTPI simply reloads the file on restart
- Changes performed through a single common routine

Implementation

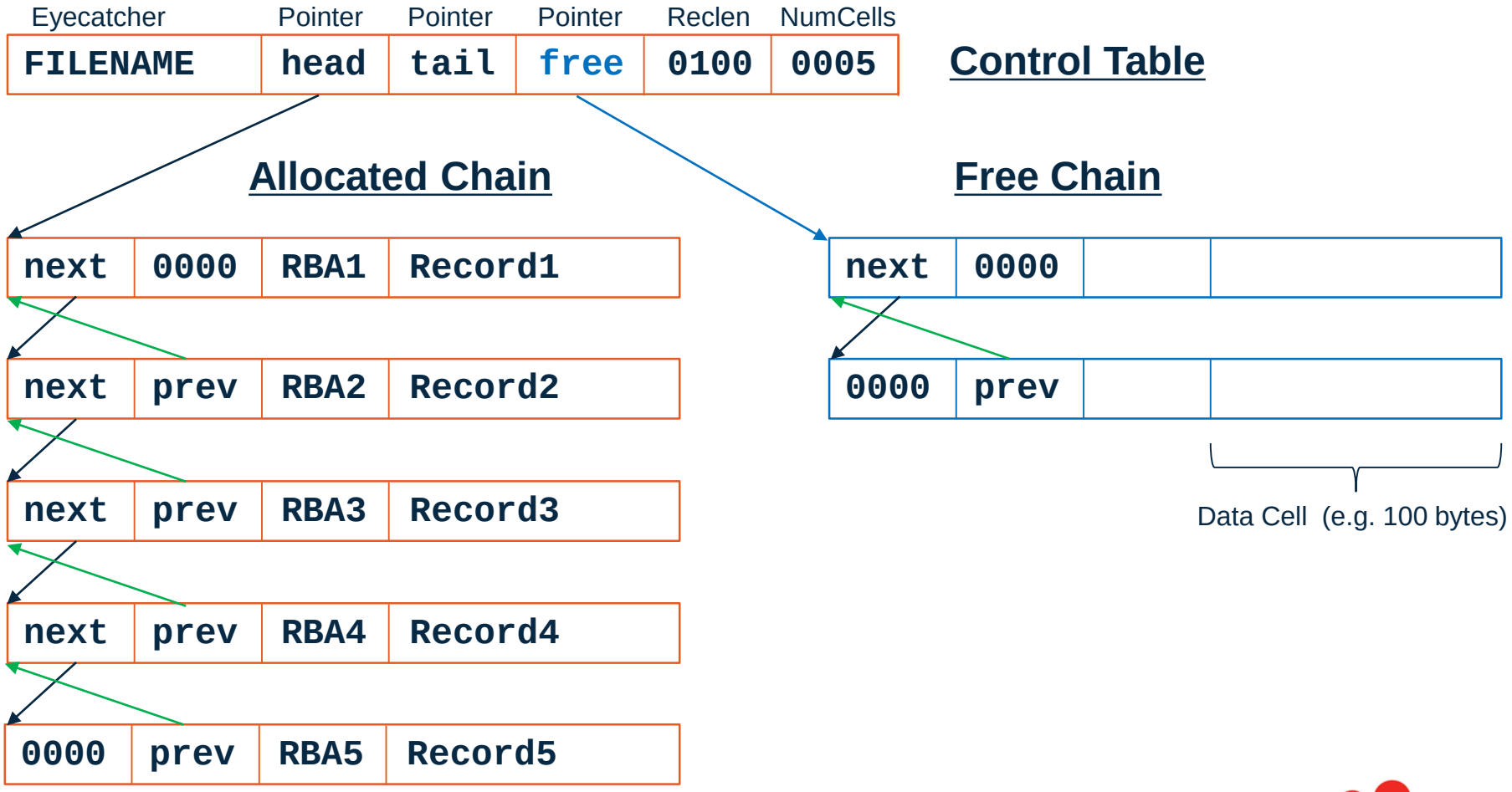
- Define a PLTPI program to LOAD the Control Table and READ all the records into the Linked List
- Each program that wants to READ the “file” can just LOAD the Control Table and chain through the Linked List
- All Updates, Additions and Deletions CALL a common subroutine to perform the function (for ESDS, not QSAM)
- Updates ENQ on the RBA, and update in place
- Additions write to the end of the file, and add the new cell to the end of the Linked List
- Deletions free the cell for subsequent Additions, and use CONTROL access on the ESDS to physically update the CI

Implementation

The following Control Table is defined for each Linked List:

```
         TITLE 'CONTROL TABLE FOR LINKED LIST OF SEQUENTIAL FILE RECS.'
FILENAME CSECT
*****************************************************************
* DEFINITION OF THE CONTROL TABLE FOR THE LINKED LIST OF RECORDS.
* IT SHOULD BE DEFINED TO CICS AS RES=YES SO IT IS NEVER FREED,
* IS LOADED ONLY AT CICS STARTUP, AND OCCUPIES ONLY 32 BYTES.
*****************************************************************
FILENAME RMODE ANY
FILENAME AMODE 31
TABLNAME DC    CL8'BLACKLST'   TABLE NAME EYECATCHER FOR DUMP
HEADPTR  DC    XL4'FF000000'   ADDRESS OF FIRST CELL IN ALLOCATED CHAIN
TAILPTR  DC    XL4'FF000000'   ADDRESS OF LAST CELL IN ALLOCATED CHAIN
FREEPTR  DC    XL4'FF000000'   ADDRESS OF FIRST AVAILABLE FREE CELL
CELLLEN  DS    F'100'          MAXIMUM LENGTH OF EACH CELL'S DATA AREA
CELLNUM  DS    F'0'            NUMBER OF CURRENTLY ALLOCATED CELLS
         END
```

Implementation



Implementation

FILENAME	head	tail	free	len	num

Then it is defined in the application program as follows:

```
LINKAGE SECTION.
01 Filename-CTRL.                    <-(For example)
   05 List-Name   PIC X(8).          <-(useful in a dump)
   05 Head-PTR    POINTER.
   05 Tail-PTR    POINTER.
   05 Free-PTR    POINTER.
   05 Cell-Len    PIC S9(8) COMP.
   05 Cell-Num    PIC S9(8) COMP.
```

Implementation

And for each Linked List, the Cell is defined as:

```
01 This-Cell.
   05 Next-PTR POINTER.
   05 Prev-PTR POINTER.
   05 This-RBA PIC S9(8) COMP.   <- for ESDS
   05 This-Data.
      10   Whatever is needed.
```

Implementation

So the program simply performs the following:

```
EXEC CICS LOAD
     PROGRAM (Filename)
     SET (ADDRESS OF Filename-CTRL)
END-EXEC
```

Do not move any values to any of the fields in Filname-CTRL. These will all be pre-initialized by the PLTPI program.

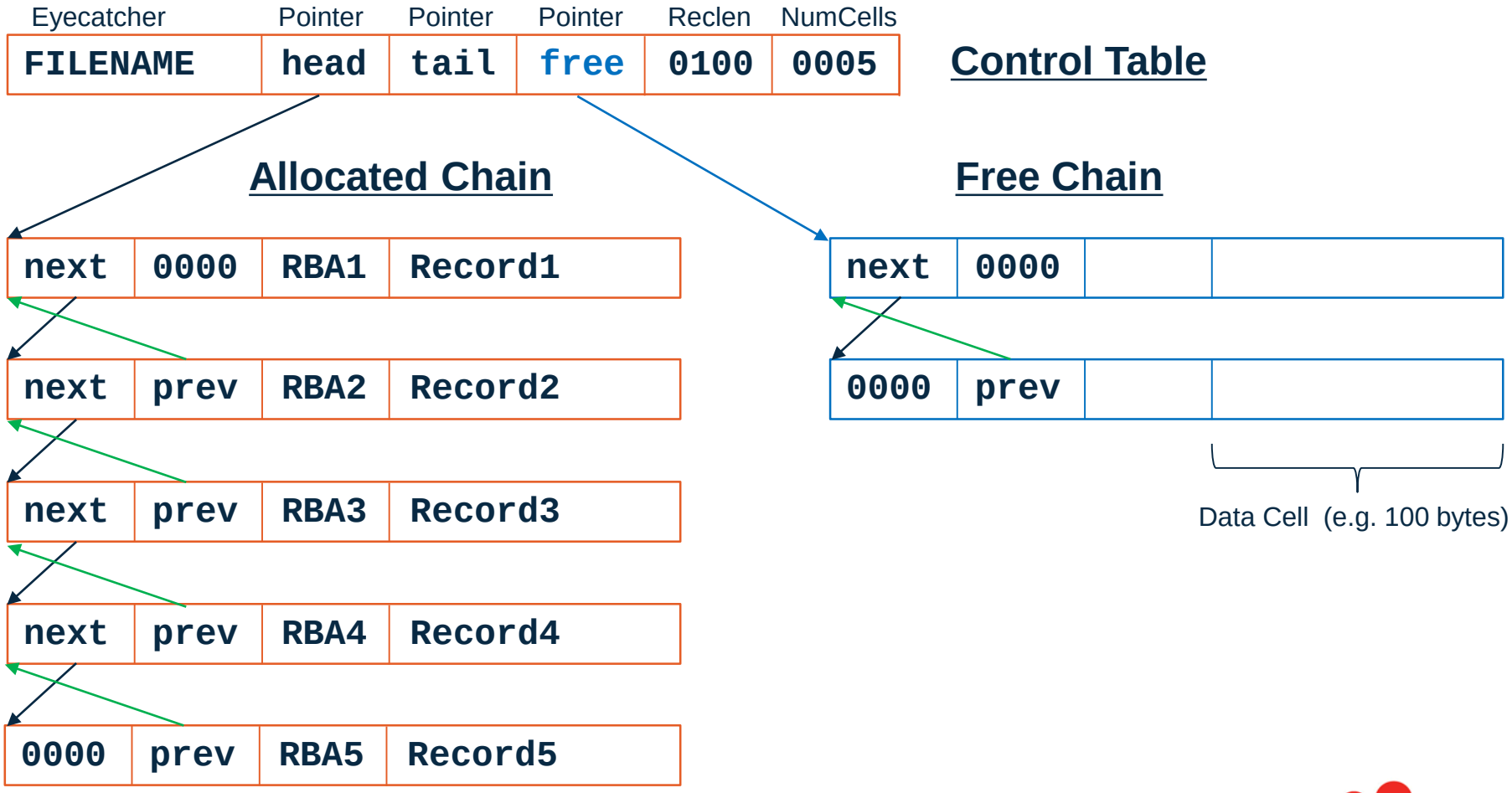
Implementation

Then “Read” and process each record as follows:

```
SET ADDRESS OF This-Cell TO Head-PTR
PERFORM UNTIL ADDRESS OF This-Cell IS NULL
      Process This-Data
      ,
      ,
      SET ADDRESS OF This-Cell TO Next-PTR
END-PERFORM
```

We can also process the List in reverse (LIFO) order by using Tail-PTR and Prev-PTR instead of Head-PTR and Next-PTR

Implementation



Refinements and Extensions

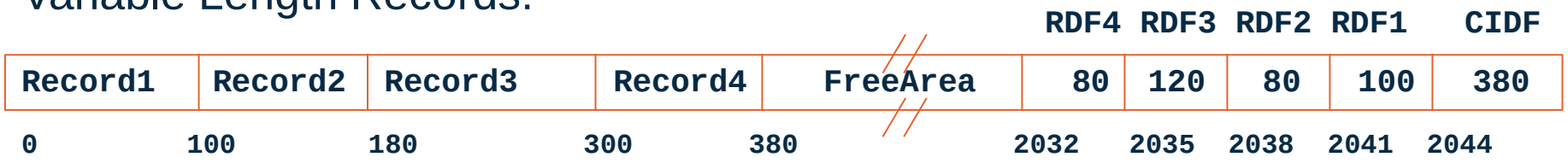
- A Browse function could display the details of 20 records at a time
- It could perform updates in place as long as the updates are single-threaded (ENQ)
- If an ESDS is to be updated then define the dataset profile with CONTROL access so CI can be manipulated directly

Refinements and Extensions

ESDS Control Interval

Fixed Length Records:

Variable Length Records:



Refinements and Extensions

- Since ESDSs are not officially recoverable, any changes must be logged if forward or backward recovery is required
- Since all records are available to all tasks (in this version), we should move our record to working-storage if we execute any CICS commands during our use of it
- If we DON'T execute any CICS commands within the loop performed for each record, then an occasional SUSPEND command would avoid a possible runaway task
- Functional Routines for WRITE, REWRITE & DELETE would all be generic to ensure Threadsafe operation

Refinements and Extensions - Summary

- Define the Linked-List Loading Program in PLTPI
- Assemble & Link this Program into the RPL
- and define as RESIDENT
- Filename is passed as Parameter to the Loading Program
- Everything is defined by the 28-byte Filename-CTRL Table
- The Application Program LOADs the Filename-CTRL Table
- and addresses the first record by using HEAD-Ptr
- Then simply moves NEXT-Ptr to ADDRESS OF This-Cell
- to access each subsequent record
- Because everyone is accessing the same record areas, this is NOT THREADSAFE! So how can we make it so?

Making it all Threadsafe

- We need to insulate each task from every other task, by ensuring that only one task at a time can access a record
- This is necessary even for READ-only, because if someone else has access to the same address, they could change it
- So copy each record to a free cell in a MAXTASK list, and pass the address of THAT cell instead of the original record

Making it all Threadsafe

We would then “Read” and process each record as follows:

```
SET ADDRESS OF This-Cell TO Head-PTR
CALL GetNext USING ADDRESS OF This-Cell
PERFORM UNTIL ADDRESS OF This-Cell IS NULL
    Process This-Data
    CALL GetNext USING ADDRESS OF This-Cell
END-PERFORM
```

Making it all Threadsafe

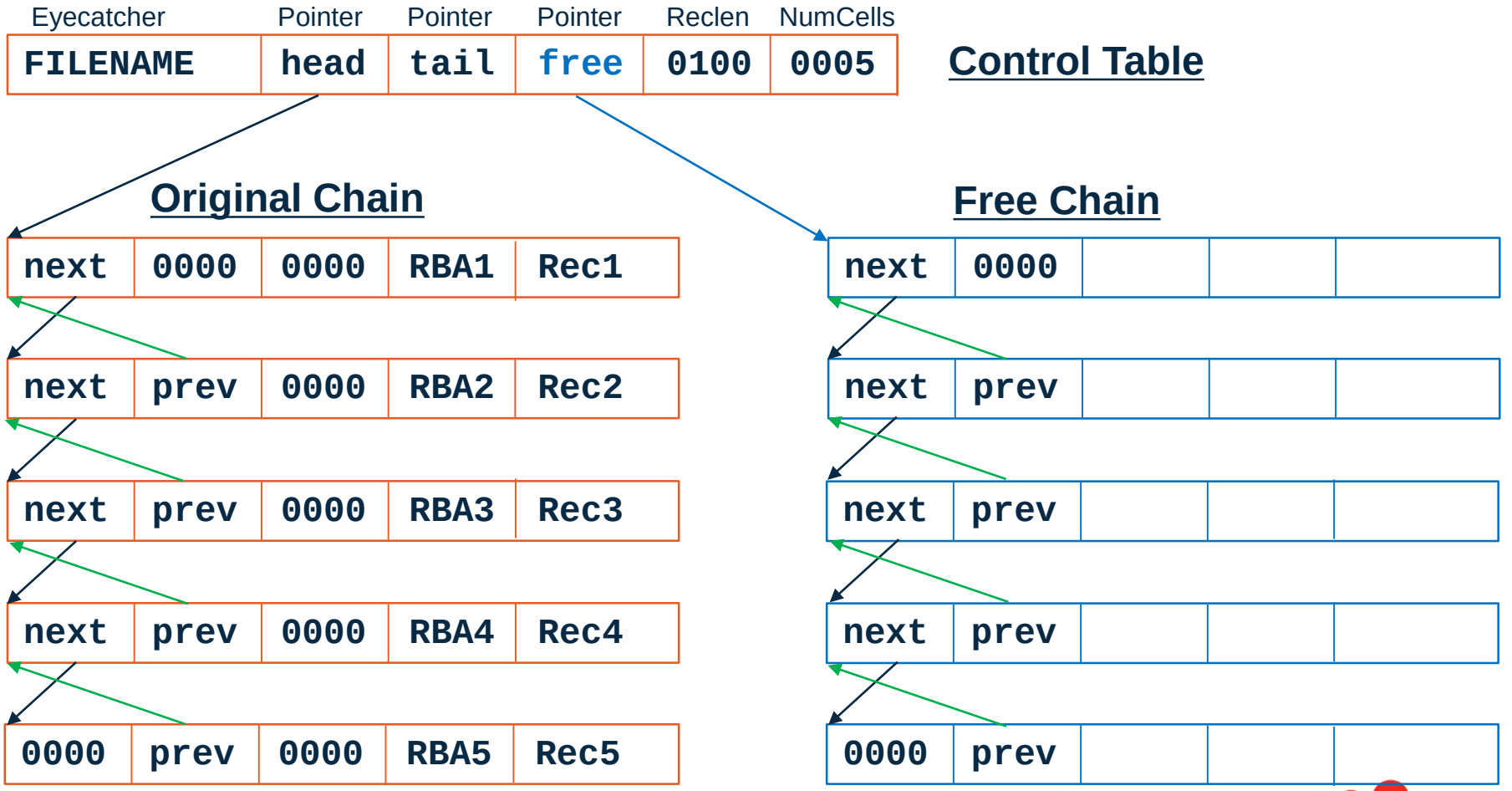
Let’s consider a situation where:

Task1 reads Record1

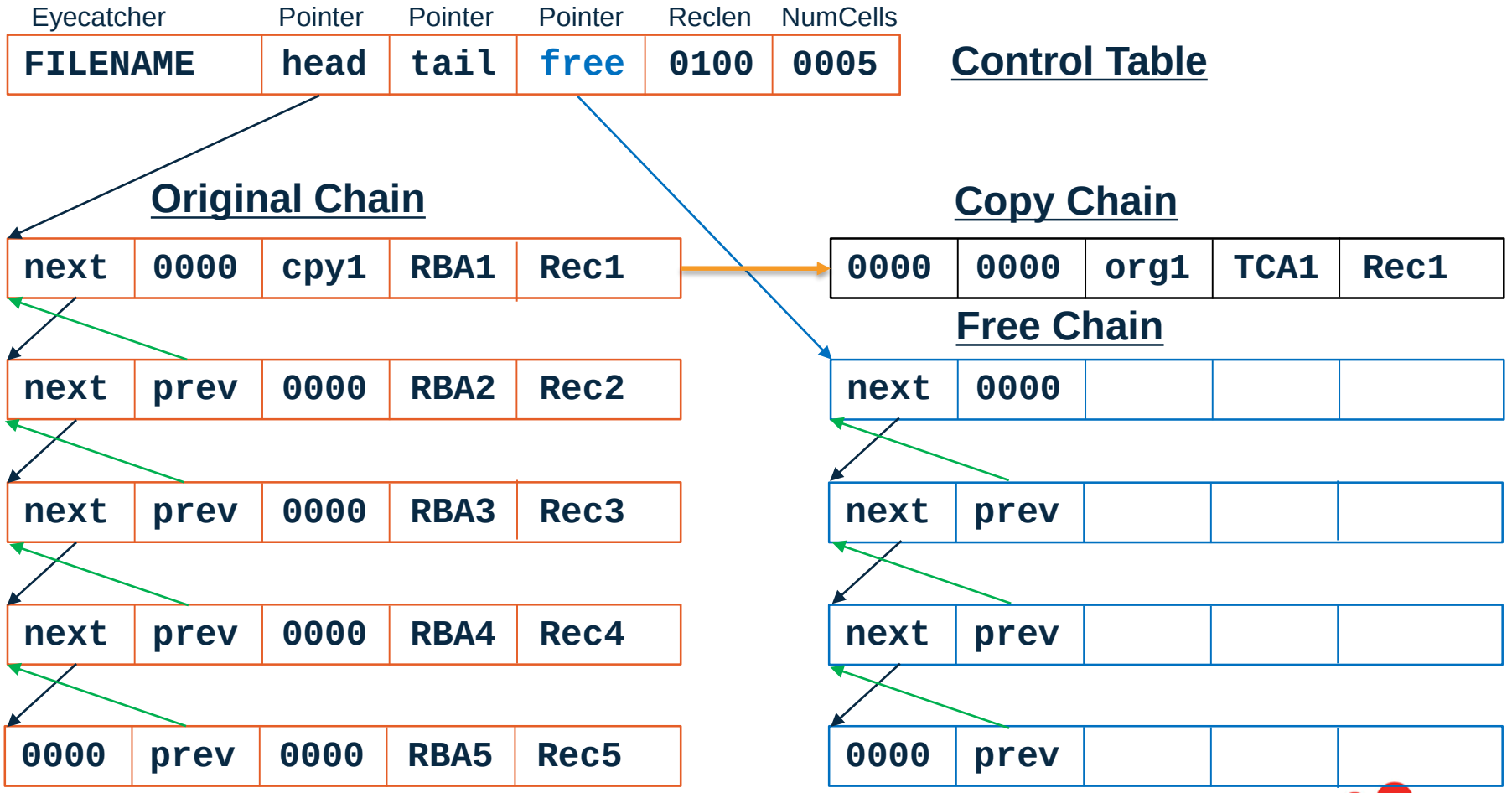
Then Task2 reads Record1

Then Task2 reads Record2

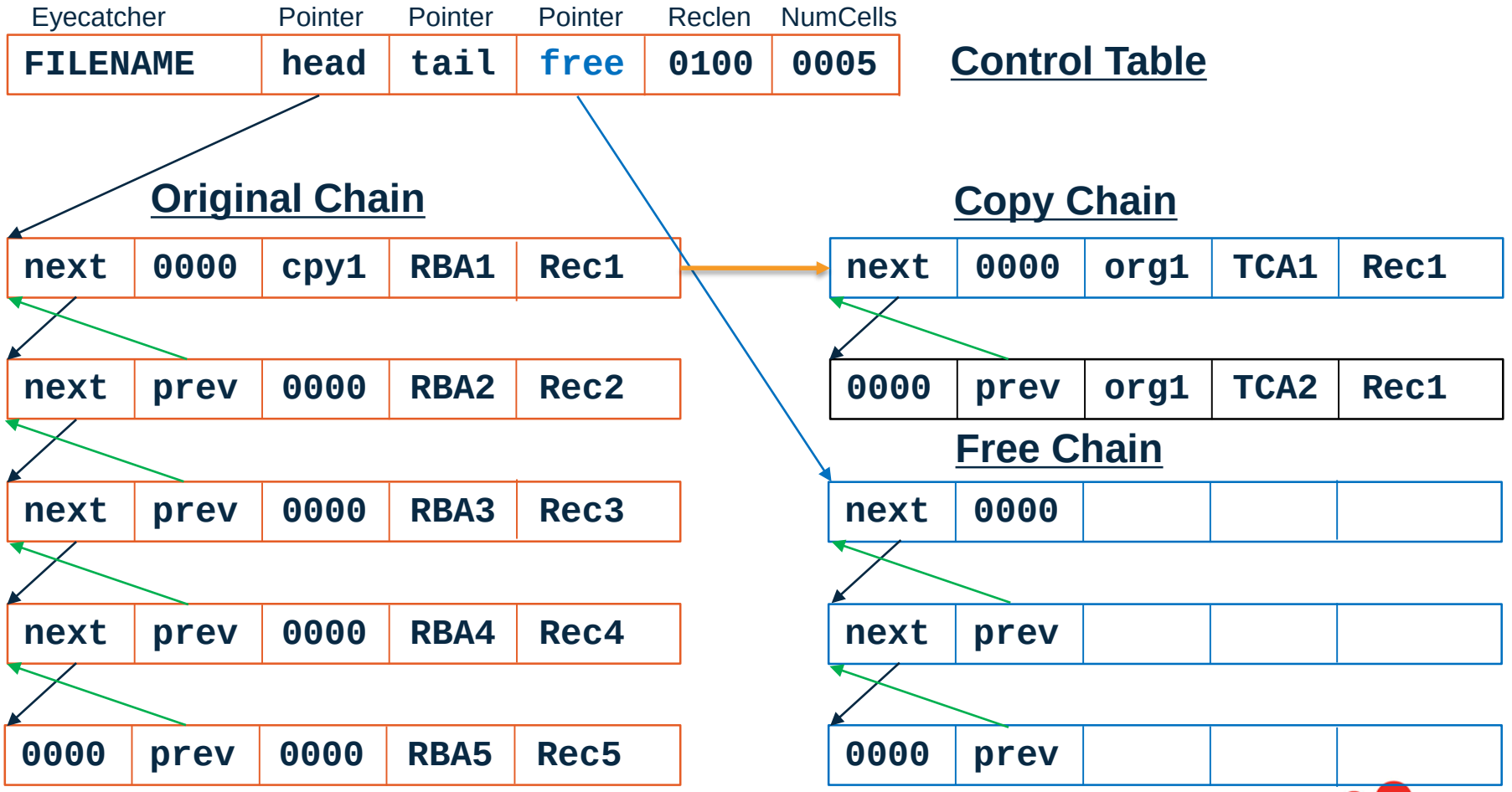
Making it all Threadsafe



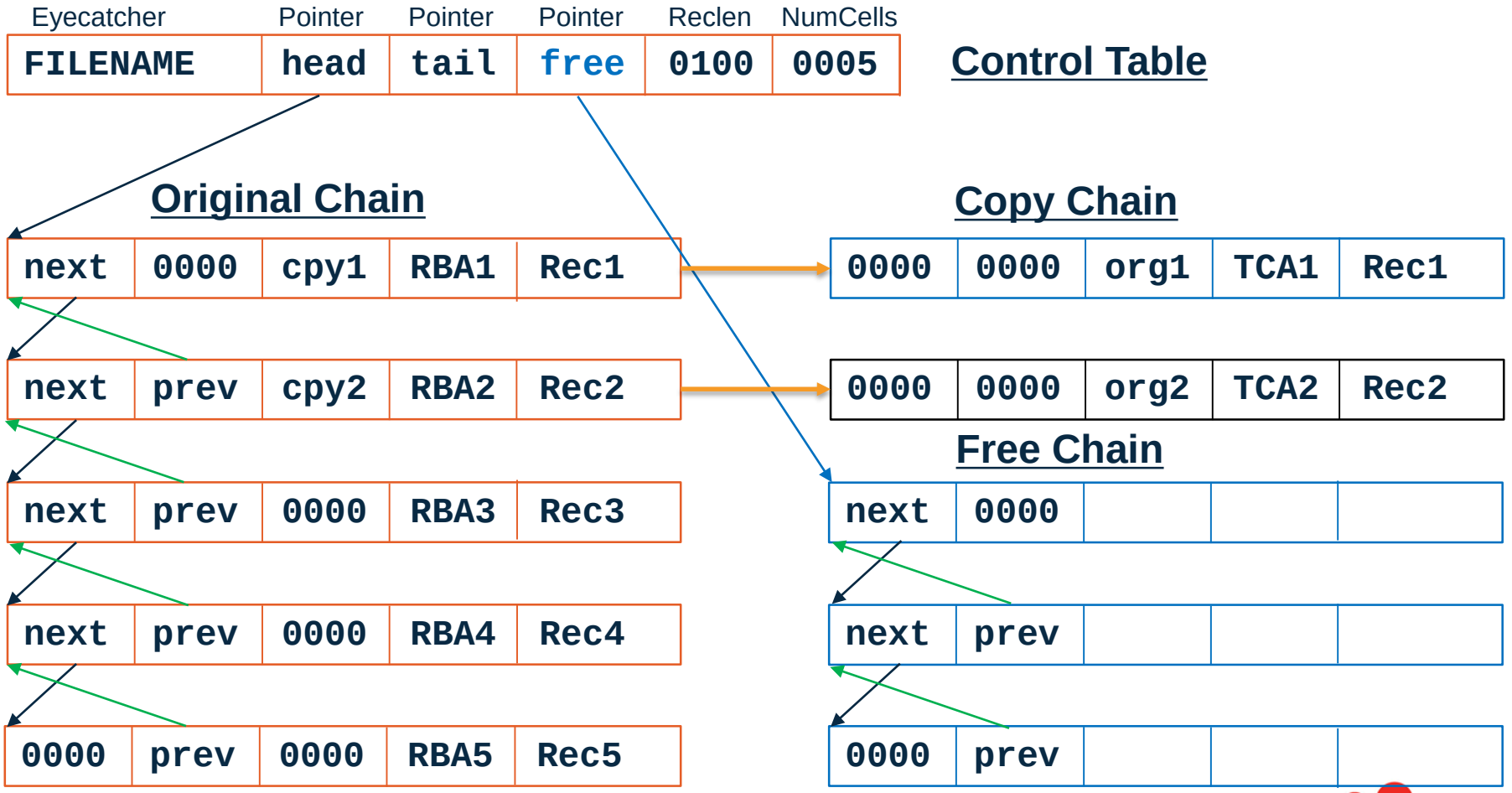
Making it all Threadsafe



Making it all Threadsafe



Making it all Threadsafe



64 Bit - A Whole New World !

- 64 bit addresses open up an address space that is
  9 BILLION times bigger than we have had until now !

- To give us a sense of what that really means, if we think of a 64 bit address space as reaching from here to the moon, how far off the ground would a 31 bit address space reach?

- **LESS THAN 2 INCHES ! ! !**

- So let's use 64 bit addressing and put the DATA above the bar. Just keep the linked list of ADDRESSES below the bar

64 Bit - A Whole New World !

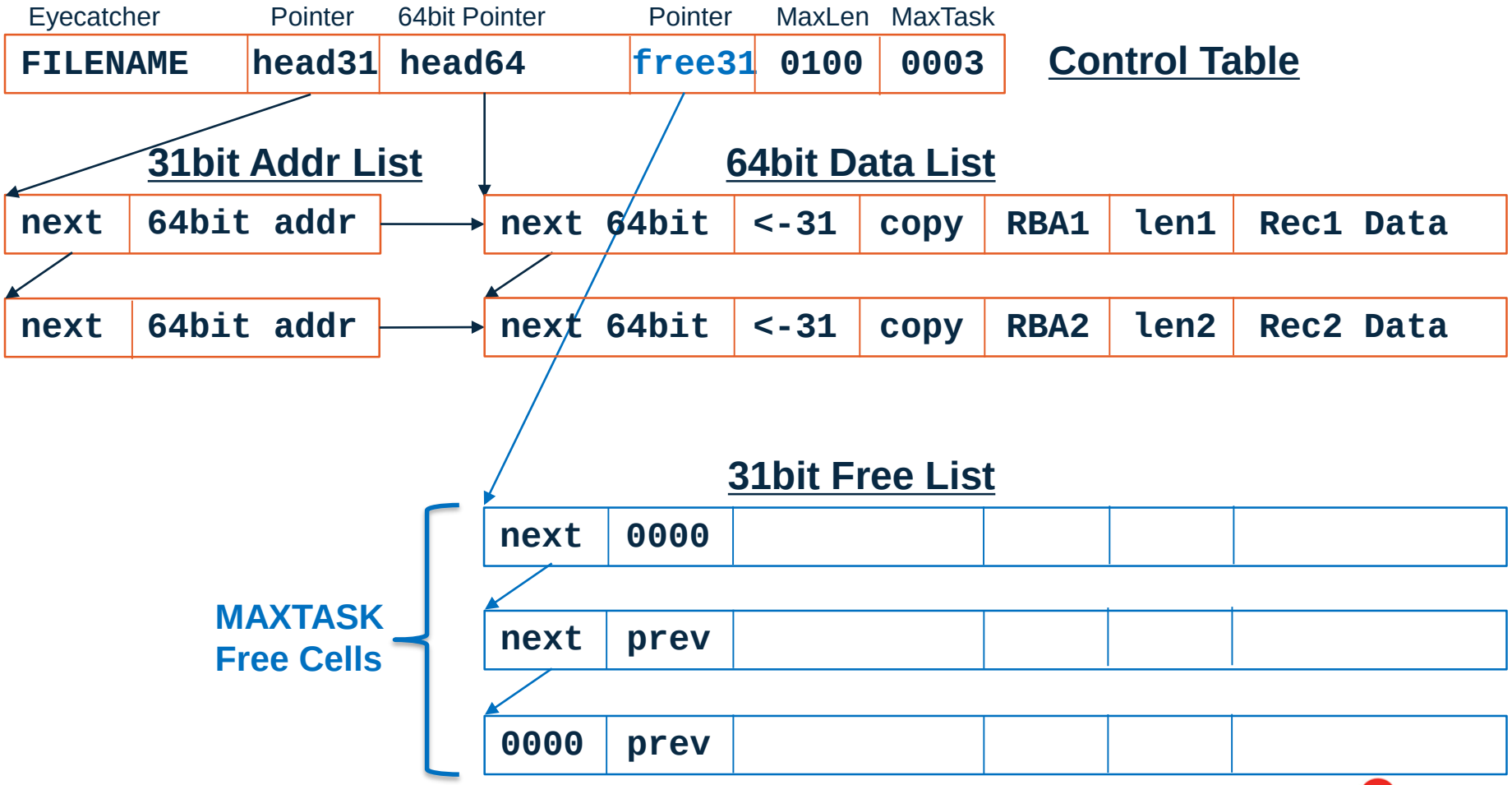
For 64 bit, the Control Table defines a Linked List of **Addresses**, and the records are moved down below the bar as required

```
01  This-Cell.
    05  Next-PTR   POINTER.
    05  This-RBA   PIC S9(8) COMP. <– for ESDS only
    05  This-Len   PIC S9(8) COMP. <– length of data
    05  This-Addr  PIC X(8).       <– 64 bit Address
    05  Curr-PTR   POINTER.        <– 0 if not below the bar now
    05  Curr-CTR   PIC S9(4) COMP. <– current # of this rec in use
```

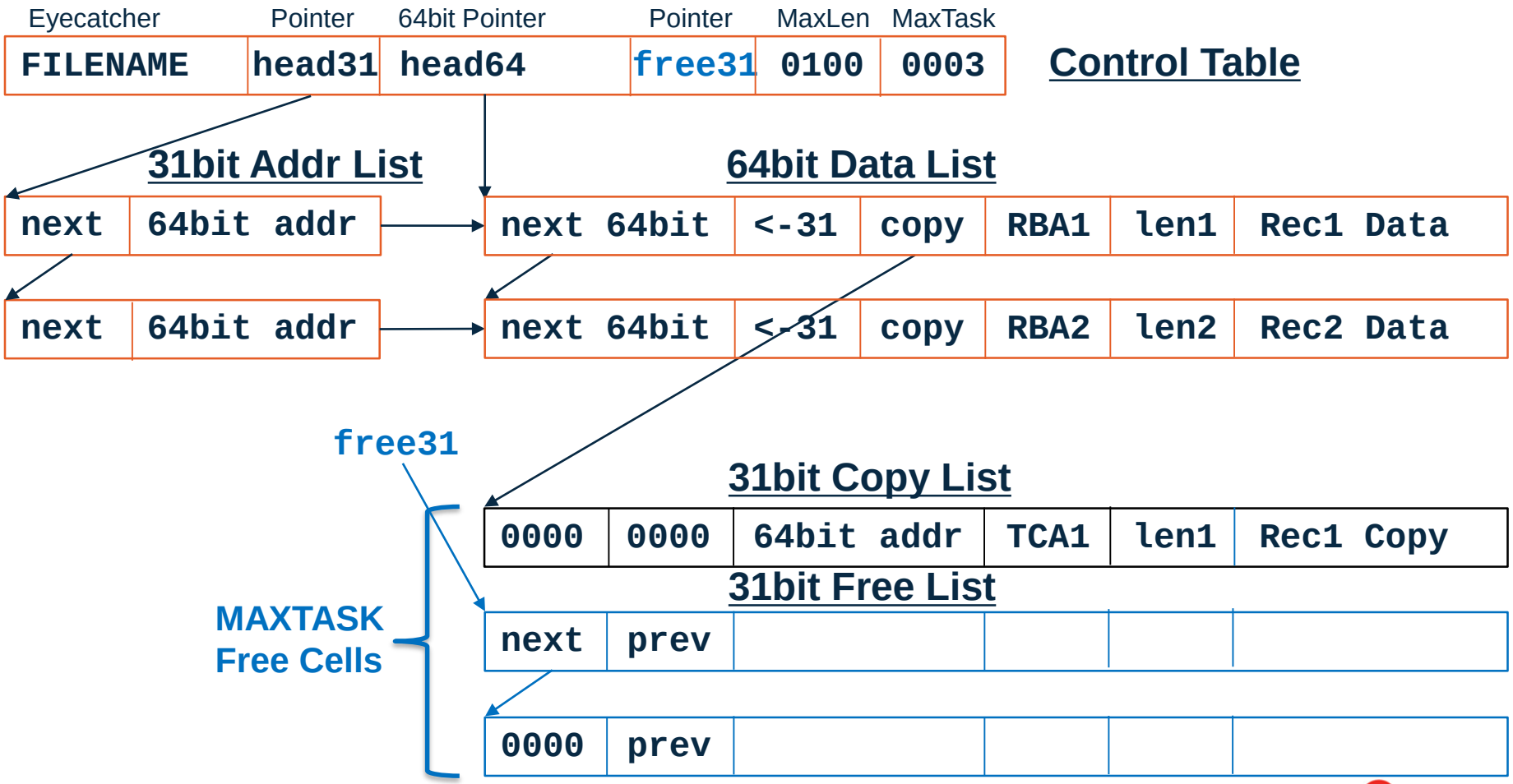
with the data defined as:

```
01  This-Data.
    05  This-Len   PIC S9(8) COMP. <– enables variable length recs
    05  Whatever is needed.
```

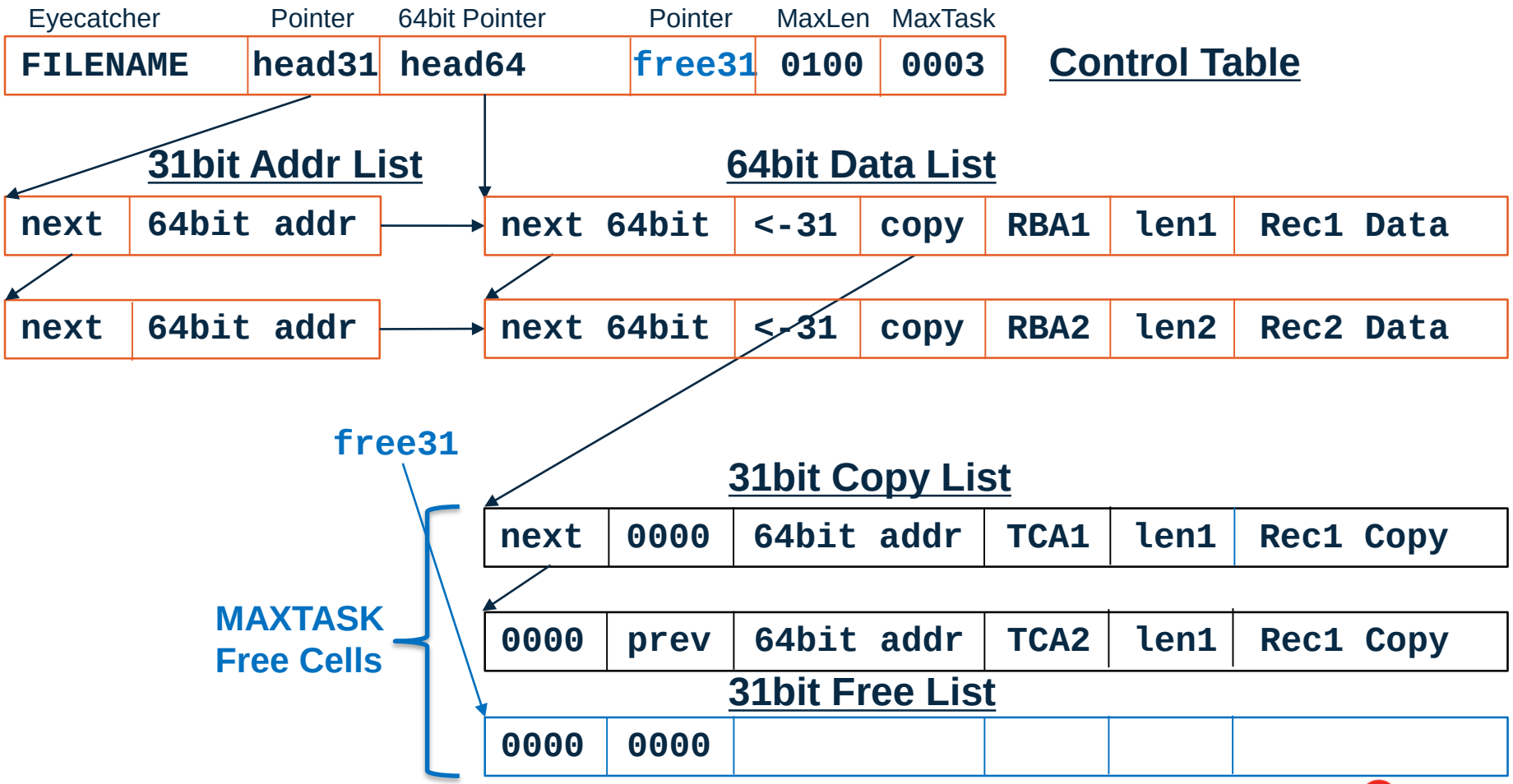
64 Bit - A Whole New World !



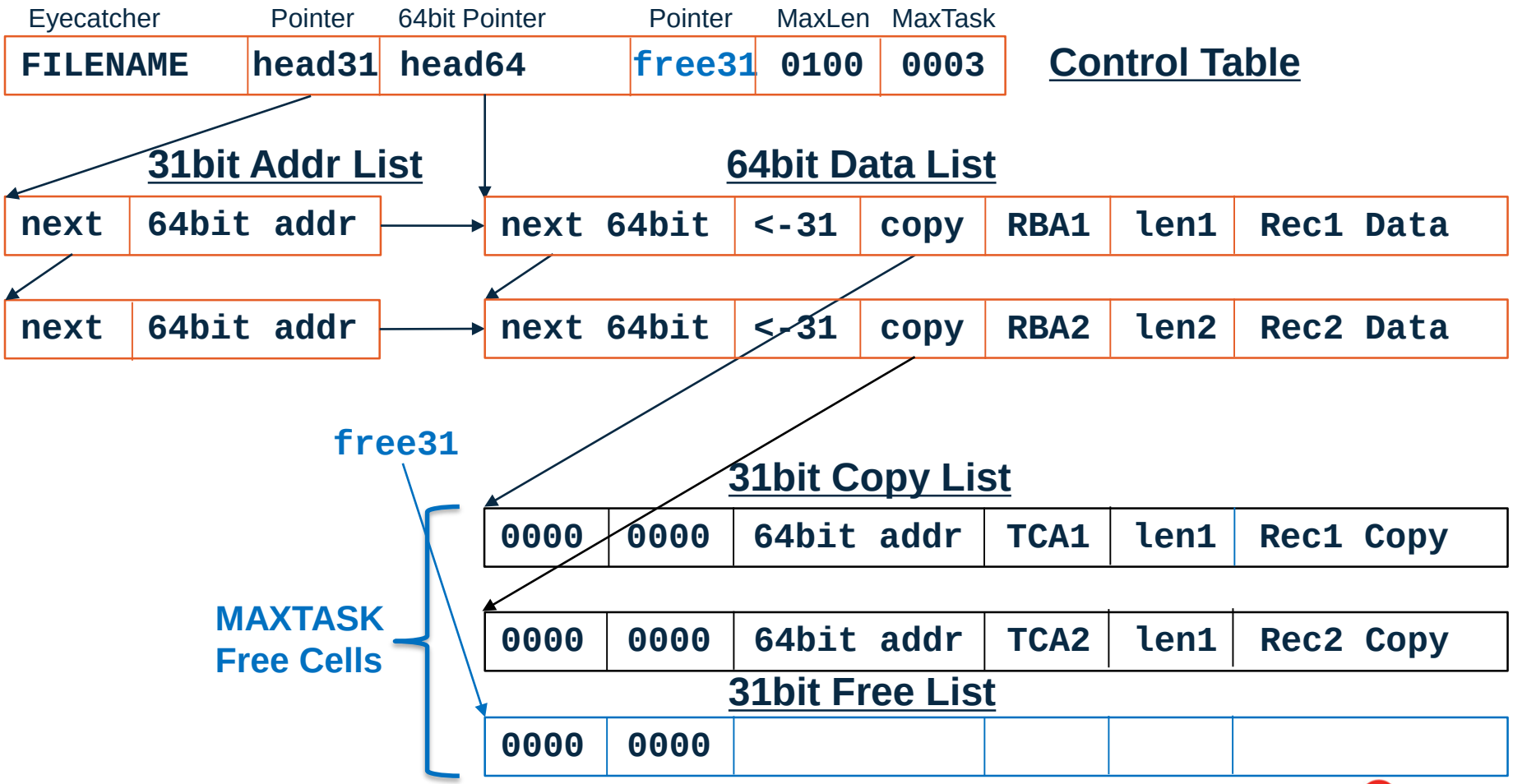
64 Bit - A Whole New World !



64 Bit - A Whole New World !



64 Bit - A Whole New World !



64 Bit - A Whole New World !

If you would like any help with any of these techniques, please call me on +61-414-SPREID or +1-925-452-6567, or email me at StephenPReid@outlook.com

Questions?