

COMP 3300 TALK

Slide 2:

The Palm OS is situated on the Palm Handhelds. Due to this constraint the OS is small and embedded on these devices.

Because the Palms have so little resources, for instance the first Palm had a 16Mhz processor, 128KB of RAM and 512KB of ROM. As there is only a small amount of memory to work with, not only does the OS have to small to fit, it also has to be efficient with memory so that it does not run out while executing programs.

Slide 4:

The Storage Ram as has been mentioned stores user data such as appointments etc, as well as some user installed programs. It can be likened to the hard drive of your normal computer – you store your programs and data on this.

Even though it is RAM it is permanent due to the fact that the Palm never really shuts down completely – however this isn't necessarily the safest thing – because the memory could be wiped if for any reason the power supply failed.

In earlier OS's such as version 1 and 2, the Storage RAM was split into 65KB storage heaps. However in the later versions all the Storage RAM was just a single storage heap.

The data in Storage RAM is all kept as records in the Data Manager, which we will talk about later, so for an application to access one of these storage heaps it, will use the Data Manager.

Slide 6:

All the data, either in Storage or Dynamic RAM or in ROM is stored in chunks. These chunks can be any size from 1KB to 64KB.

Each chunk is stored in heap. These heaps are referenced through heap ID's, with the first heap on card 1, having the id 1 and so on, then through memory card 2 and so on as well.

Each chunk is referenced by what is called the local ID. These are just offsets from the base address of the card the chunk is on.

A chunk can be movable, or non-movable – obviously any chunk that is in ROM is non-movable. Programs access each of these types of chunks differently.

The non-movable chunks are just accessed by a pointer – which remains valid, as the chunk never moves. Therefore the local ID for a non-movable chunk is just the offset from the base to the chunk.

However the access of a movable chunk is a lot more complex. A pointer to the chunk is still stored, called the master chunk pointer – however it is not given to any application that needs to access this memory. Instead it is stored in what is called the master pointer table.

The master pointer table is used to keep track of movable chunks, and when a chunk is moved the master chunk pointer is updated in this table.

An application is given a pointer to this master chunk pointer, which does not move. This pointer is called a handle. This means that the application just uses a consistent pointer – not knowing that the actual chunk could be moving about.

The local ID therefore for a movable chunk is the offset from the base address of the card till the address of the master chunk pointer. However this ID also has the lower bit set – so that the ID's can be distinguished.

Using pointers this way does have one slight disadvantage – performance, it is obviously slightly slower than using a direct pointer.

Slide 8:

The Data Manager as mentioned stores each chunk of Storage RAM as a record in a database. This database on the Palm is equivalent to a file on a desktop system. For instance they can be created, opened deleted and closed by application.

Each of these records in is actual fact a chunk that is controlled by the memory manager. That means for the Data Manager to actually control its records it calls the memory manager.

These records can be stored anywhere in RAM, however each record belonging to the same database must all be on the same card.

As an example, if a program wishes the access something in storage RAM it calls the Data Manager to retrieve the record from the database. The Data Manger will fetch the local ID of the record from its header and

will then either calculate the handle of the record if the record is movable (it can tell due to lower bit being set) or just return the direct pointer.