Reliable LoadExecs

Topics Covered

- LoadExec Explained
  - What is LoadExec
  - Advantages
  - Uses
  - Internally
- Disadvantages of LoadExec
- Alternatives to LoadExec
  - Overlays
  - Load then Exec
- Q&A

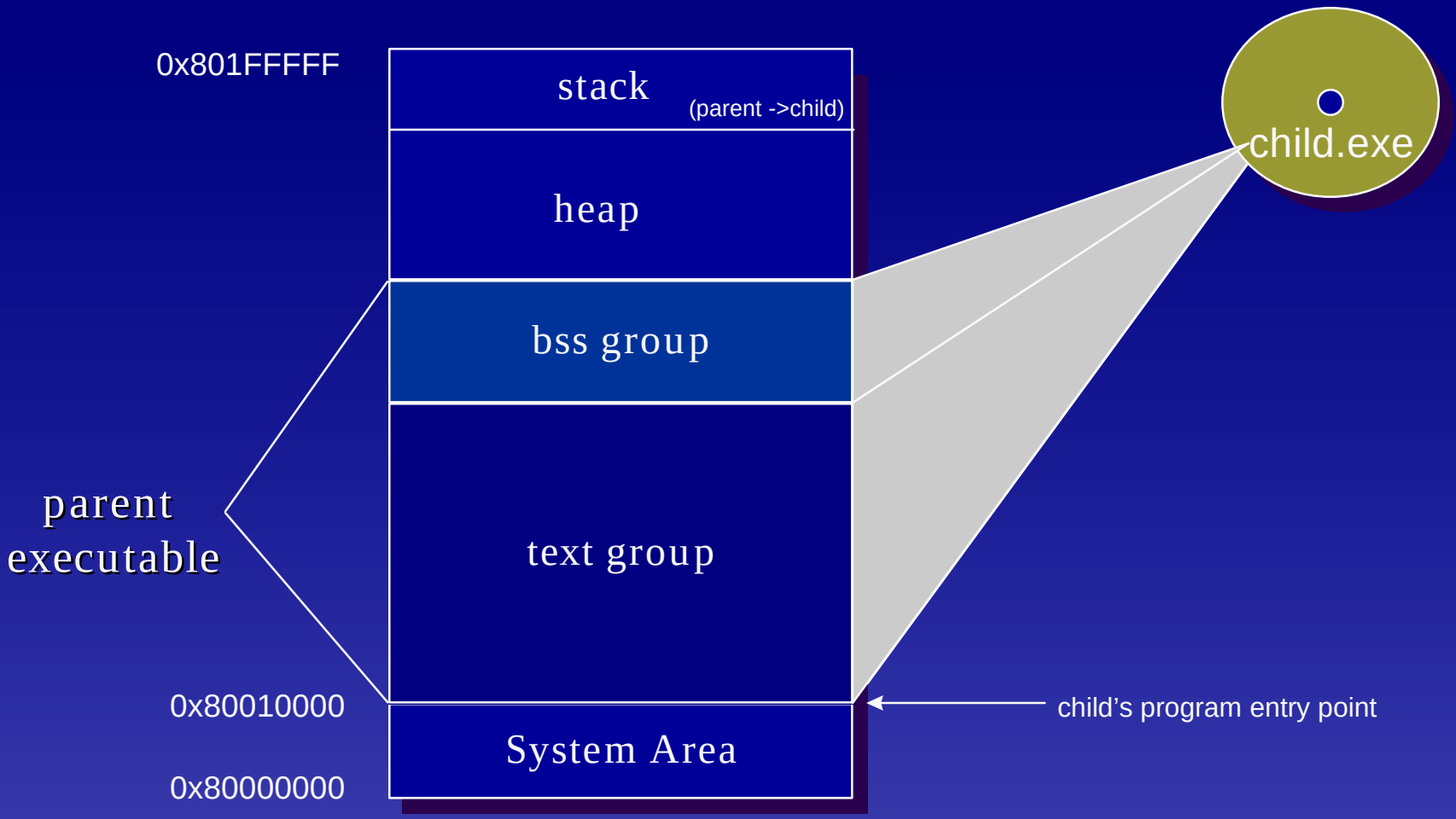
LoadExec Explained

What is Load Exec?

- LoadExec is a Libapi function that loads an executable from CD into main memory and jumps to the program entry.
  - Calls Load and Exec Internally
  - Executes from ROM
  - Loads child over parent

```
long LoadExec( name, s_addr, s_size )
char *name; /*executable name*/
unsigned long s_addr, s_size;/*stack vals*/
```

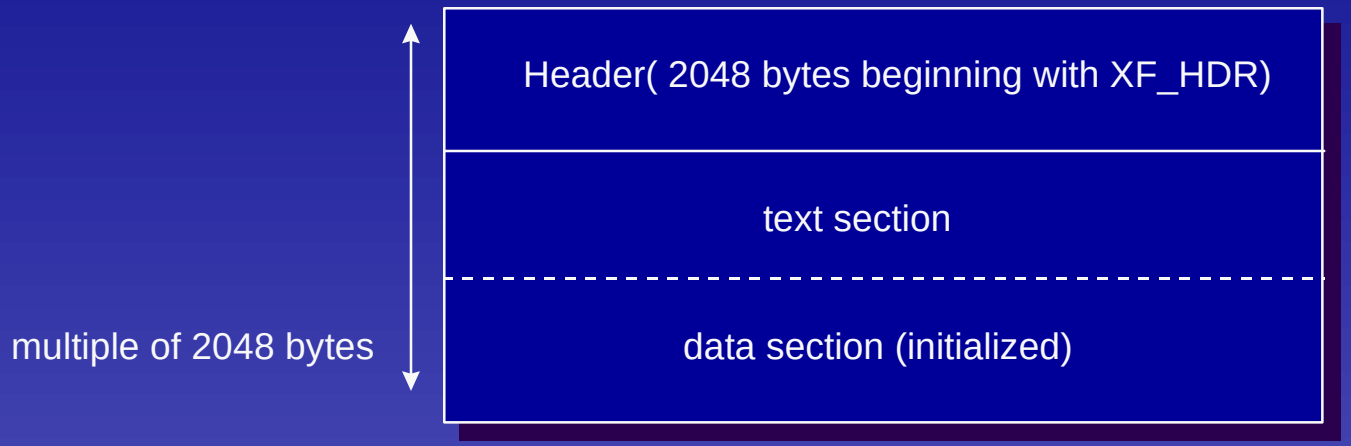
What is LoadExec?



What is LoadExec?

- PSX Executable format (as defined in kernel.h)

```
struct XF_HDR {
        char key[8];            /* key code*/
        unsigned long text;     /* size of text sectioin*/
        unsigned long data;     /* size of data section*/
        struct EXEC exec;       /* executable info*/
        char title[60];         /* license code */
};
```



What is LoadExec?

- PSX Executable format (as defined in kernel.h)

```
struct EXEC {
        unsigned long pc0;          /*initial value of the program counter*/
        unsigned long gp0;          /*global pointer*/
        unsigned long t_addr;        /* start address of the text group*/
        unsigned long t_size;       /*size of the text group*/
        unsigned long d_addr;       /*system reserved*/
        unsigned long d_size;       /*system reserved*/
        unsigned long b_addr;       /*start of the bss group*/
        unsigned long b_size;       /*size of the bss group*/
        unsigned long s_addr;       /*lead address of the stack*/
        unsigned long s_size;       /*stack size*/
        unsigned long sp,fp,gp,ret,base; /*register shunt area*/
};
```

Advantages

- Allows applications to overcome the 2 MB barrier
- Child processes may use all of Memory
- Allows parallel development
- Easy to implement

LoadExecs Uses

- Demos / Samplers
- Multi-disc titles
- To move between exclusive segments of a game
  - Intro -> game
  - Levels
  - Cinematic sequences

LoadExec Internally

- Saves the current stack values
- Calls ExitCriticalSection()
- Calls Load()

```
addr = exec.s_addr;
size = exec.s_size;
ExitCriticalSection();
if(Load(fn,&exec)==1)
```

LoadExec Internally

- Load()
  - Fills in system maintained EXEC struct *exec*, with the requested executable's information.
  - Loads text group to *exec->t_addr*
  - Calls FlushCache()
    - Bad to do outside of a critical section

LoadExec Internally

- If Load() fails…
  - Attempts to Load() default executable
- If Load() is successful…
- Calls Exec()
  - Passing the address of the system EXEC *exec* as an argument

LoadExec Internally

- Exec()
  - Saves s*p*, *gp*, *fp*, *ra* registers to shunt area in *exec*
  - Clears bss
    - If *exec->b_size != 0*, the programs bss group is cleared using *exec->b_addr* and *exec->b_size*
  - Sets-up stack
    - If *exec->s_addr* != 0, set *sp* and *fp* to *exec->s_addr* + *exec->s_size*
  - Shifts *a2->a1->a0* to allow retrieval of arguments
  - Pushes *exec->pc* to a temp reg and does a *ja* on that reg

Disadvantages of LoadExec

Disadvantages of LoadExec

- When invoking:
  - All interrupts must be stopped
  - Must call _96_init()
    - slow
- Difficult to pass data between processes

Disadvantages of LoadExec

- Once called all control is relinquished
  - No error handling
    - Load() must successful on first try
      - read() done within in Load also only gets one shot
  - No returning to parent
  - Not because parent is overwritten, but…

Disadvantages of LoadExec

- If Load() fails on the requested executable...

  and...

  Load() fails on the default executable...

- LoadExec enters an infinite loop!!!
  - while(1) { }

Disadvantages of LoadExec

- The Moral of the Story is…

There is no 100% reliable LoadExec()

Alternatives to LoadExec

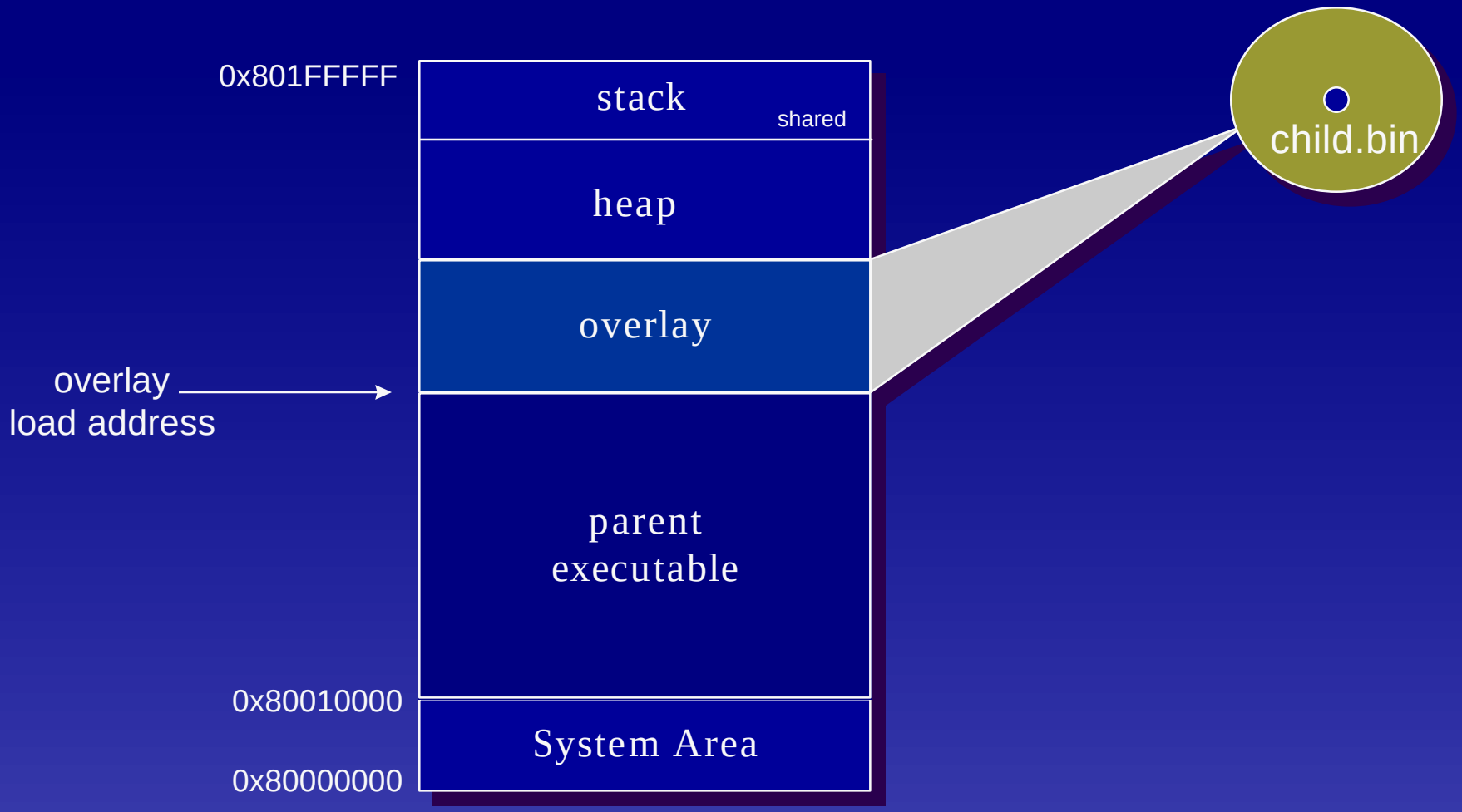
Alternatives to LoadExec

- Overlays
- Load then Exec

Overlays

- A method of linking portions of your code and data to the same memory location then swapping them in and out as needed.

Overlays


0x801FFFFF
stack
shared
heap
overlay
overlay
load address
parent
executable
0x80010000
System Area
0x80000000
child.bin

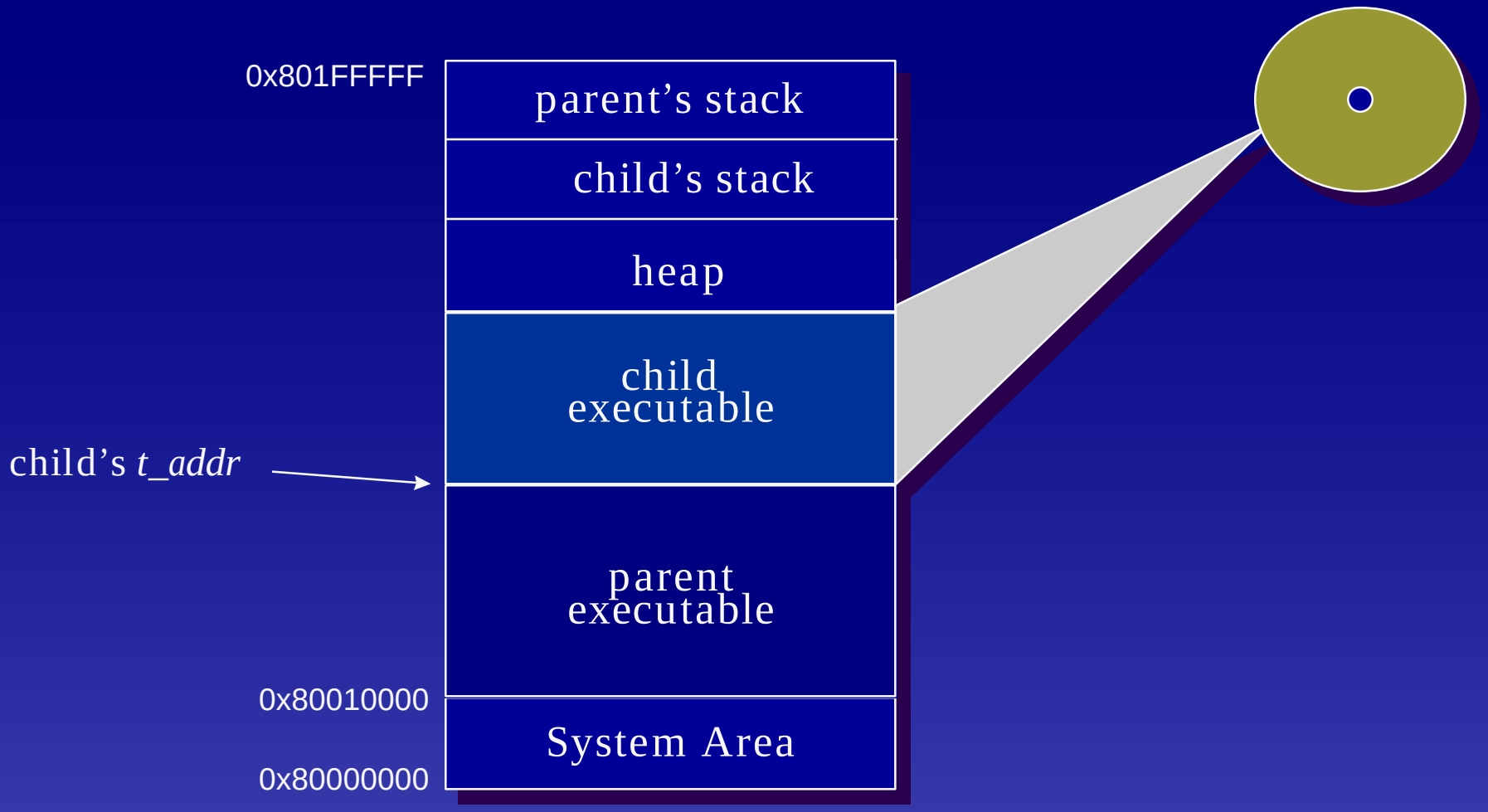
Overlays

- Advantages:
  - Not necessary to stop interrupts.
  - A separate process can be executed while the child is loading.
  - Can pass arguments and share data.
- Disadvantages:
  - The make operation is more complex.
  - The two processes must share memory.
  - Processes must be exclusive.

Load then Exec

- Application calls Load() to get executable from CD into main memory, then calls Exec() to pass control to that executable.

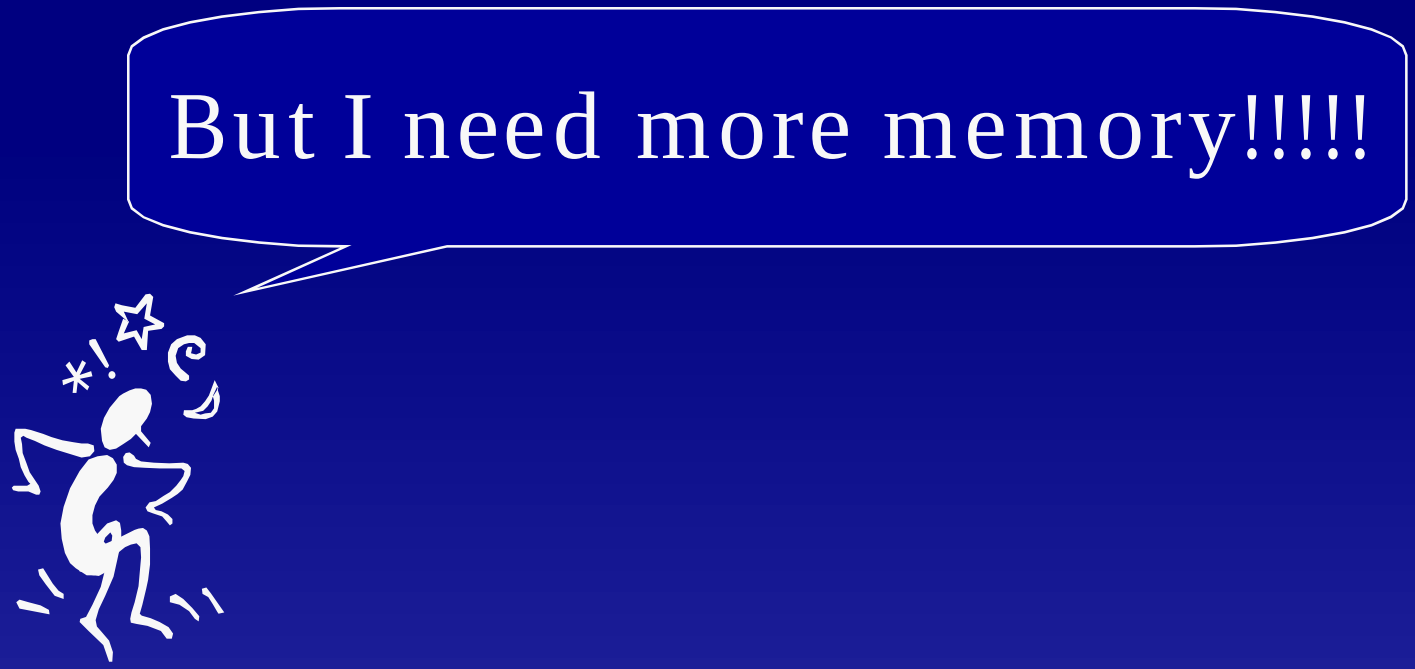
Load then Exec



Load then Exec

- Advantages:
  - Allows for parallel development
  - More control
    - Allows for error handling
    - Application can use own loading scheme i.e. CdRead
  - Return to parent is possible
  - Easier data sharing
- Disadvantages:
  - Interrupts must be stopped
  - Two copies of Libraries
  - Parent and child must operate in limited memory

Load then Exec



- Implement loader scheme
  - Small piece of code that exist only to load and launch larger modules.

Load then Exec - Loader

- Loader
  - small
  - linked just after System Area or in other "safe" region
- Child modules
  - Link just beyond loader
  - Link with none2.obj and explicitly set stack below Loader's stack

Tips...

- Parent exit procedures
  - ResetGraph(0) or (3)
  - CardStop()
  - PadStop()
  - StopCallback()
  - CloseEvent()
  - _96_init
  - LoadExec() or Load()

Tips...

- If Load()
  - _96_remove
  - EnterCriticalSection()
  - FlushCache()
  - Exec()
- Child start up procedures
  - ResetCallback()
  - CdInit()
  - InitPAD()

Tips...

- If using Load then Exec
  - Repeat child start-up procedures upon return to the parent
- LoadTest(char *name, EXEC exec)
  - Loads EXEC portion of executable header
  - Returns 0 in case of failure
  - Call prior to Load() or LoadExec()
    - retry if failure
    - if retries unsuccesful inform user to change or clean disc

Tips...

- Burn CD’s at single speed

Q & A

The End