My Gold Disk Doesn't Work!

Why doesn't my Gold Disk Work?

- Aim of presentation.
  - Present methodology for debugging problems with gold disks.

Stage1: Using the Dev Kit (How?)

- Test the disk on the development system using the optional CD player.
- If it doesn’t work use the CD as a data disk and run the cpe from the hard disk e.g. use selcd rather than selemu
- If this fails then there is a problem with the data files on the disk. Either the files are named incorrectly or are in the wrong place

Stage1: Using the Dev Kit (Why?)

- Can get debug information using printf
- Can update CPE quickly without burning a new disk
- The Dev Kit is more resilient

Stage2: Testing with CDEXEC

- Cdexec is a program that attempts to run the executable from a disk on the dev kit
- Cdexec will try to run PSX.exe
- If no PSX.EXE is found SYSTEM.CNF is parsed and the executable name will be found from there

Stage2: Reasons for CDEXEC failing

- The exe was not made using CPE2X
- The system.cnf file is incorrect

Stage2: Notes on System.cnf file

- Example system.cnf
  - BOOT=cdrom:\SCUS_941.61;1
  - TCB=4
  - EVENT=10
  - STACK=801FFFF0
- Note that this information is case sensitive
- System.cnf should be in the root
- For more information see DEVGUIDE.DOC on the technical reference cd

Stage 3: OK so now we have a disk that works on the Dev Kit

- We know that the debug station is finding the executable and attempting to run it
- The exe loads and then the machine crashes

Stage 3: Why does an exe that works on the Dev Kit fail on the Debug station

- What are the differences between a dev kit and and a blue Playstation
  - The Debug only has 2Mb of ram
  - The Dev Kit has extra hardware that can be accessed
  - Mystery Crash bugs

Stage 3: Memory

- Any memory access over 2Mb will cause the machine to die
- Check the map file
- Note that the stack is also in that 2Mb

Notes on Memory

- Quite simply using more memory than is available in the target box
- Overwriting the stack with data or vice versa
- On the development system the stack pointer defaults to 8Mb and is 32K in size
- On the PS the stack pointer defaults to 2Mb and is 32K in size

On the development kit you can….

- Set the stack size by accessing the variable _ramsize;
- Set the stack size by accessing the variable _stacksize;

Notes on the stack:

- The stack can be bigger than the stack size variable indicates. The stack size is not checked by the stack allocation functions within the cpu
- 32Kb is a huge area for the stack so if you get desperate…..
- Also note on the development system, setting the stack pointer does not stop you from accessing addresses over 2Mb
- Stack grows down from the top of memory. Heap grows up from end of your program

Stage3: Dev Kit only functions

- pollhost, Pcinit, Pcopen, Pclseek, Pcread, Pcwrite, Pcclose and PSYQpause will all cause a Debug station to crash
- Printfs are OK!

Stage 4: Mystery Crash Bugs

- This is where the real fun begins:
- Something in your code is crashing, but it works fine on the development system
  - Very often this is due to initialization problems
  - Don’t assume an unassigned variable got assigned to 0

Stage 4: How to debug on the Debug station

- Typically program crashing in initialization
- Conventionally can't get printf information back from target
- FntPrint no good
- Find a work around
  - e.g.: Use Clearimage() to clear the display buffer between instructions
- This is time consuming, painful and expensive

Stage 4: How to debug on the Debug station (notes)

- init_libs();
- clear_screen_and_pause(RED);//if we got a red screen we got to here
- init_game();
- clear_screen_and_pause(BLUE);//if we got a blue screen we got to here
- etc

Stage 4: Use the Yaroze Cable

- use libsio
- AddSIO(BAUDRATE);
- Use a terminal program to view prints from the debug station

Correct Initialization(1)

- When not using multi-tap or gun
  - ResetCallback(0);
  - CdInit();
  - ResetGraph(0);
  - InitPAD(...);
  - StartPAD();
  - :
  - InitCARD(1);
  - StartCARD();

Correct Initialization(2)

- When using multi-tap or gun
  - ResetCallback(0);
  - CdInit();
  - ResetGraph(0);
  - InitTAP(...); / / or InitGUN();
  - StartTAP(); / / or StartGUN();
  - :
  - InitCARD(0);
  - StartCARD();

Faulty Disks

- Occasionally a disk will have errors on it caused by a damaged disk or a faulty burner
- How to check
  - Use the Verify Option on Cdgen.
  - Do a byte level Comparison of the Disk and source data
  - If unsure, burn another disk….

Faulty Disk (notes)

- Verify option on Cdgen is unstable especially with the new PCI SCSI cards It often reports non existent errors
- Errors in executable files are far more critical than errors in data

Causes of Faulty Disks

- If your cd burner has been subjected to extremes of temperature or vibration it will not write correctly and will need recalibrating
- Burners should be stored away from big changes in temperature and vibration
- Disks damaged. Disks should be stored away from bright light and extremes of temperature

Warning

- Don’t write on the surface of a gold disk with a marker pen unless is has an area designed for this… Some of the cheaper gold disks without screen printing on them can be eaten alive by the solvents in magic markers….

Tips for Happy Development

- Get loads of disks, you'll need them.
- burn disks regularly though the project lifecycle. Trap potential problems early
- Keep the file organization on the CD emulator and CD generator identical to make cutting disks simple
- Read the QA guidelines and make sure your product meets them

Additional Information

- on making disks can be found on the web site and on the developers CD:
  - see devguide.doc
  - see qa.doc
- Both these files have useful information on using creating gold disks for mastering purposes