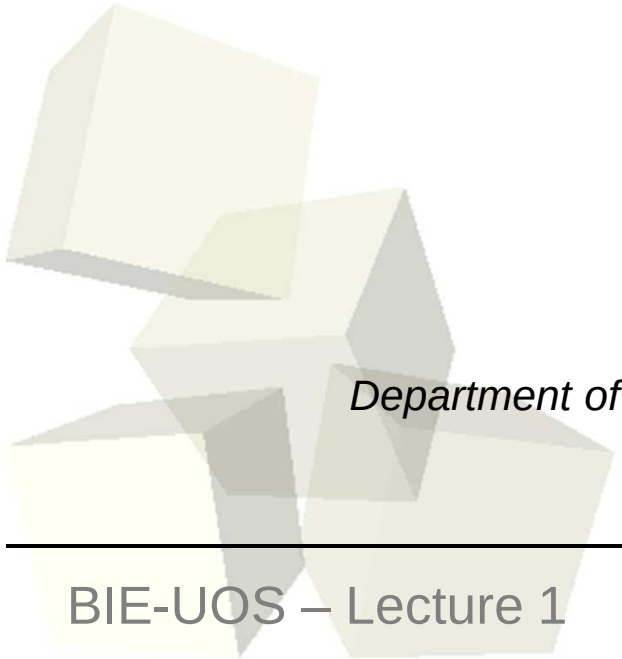
Lecture 1

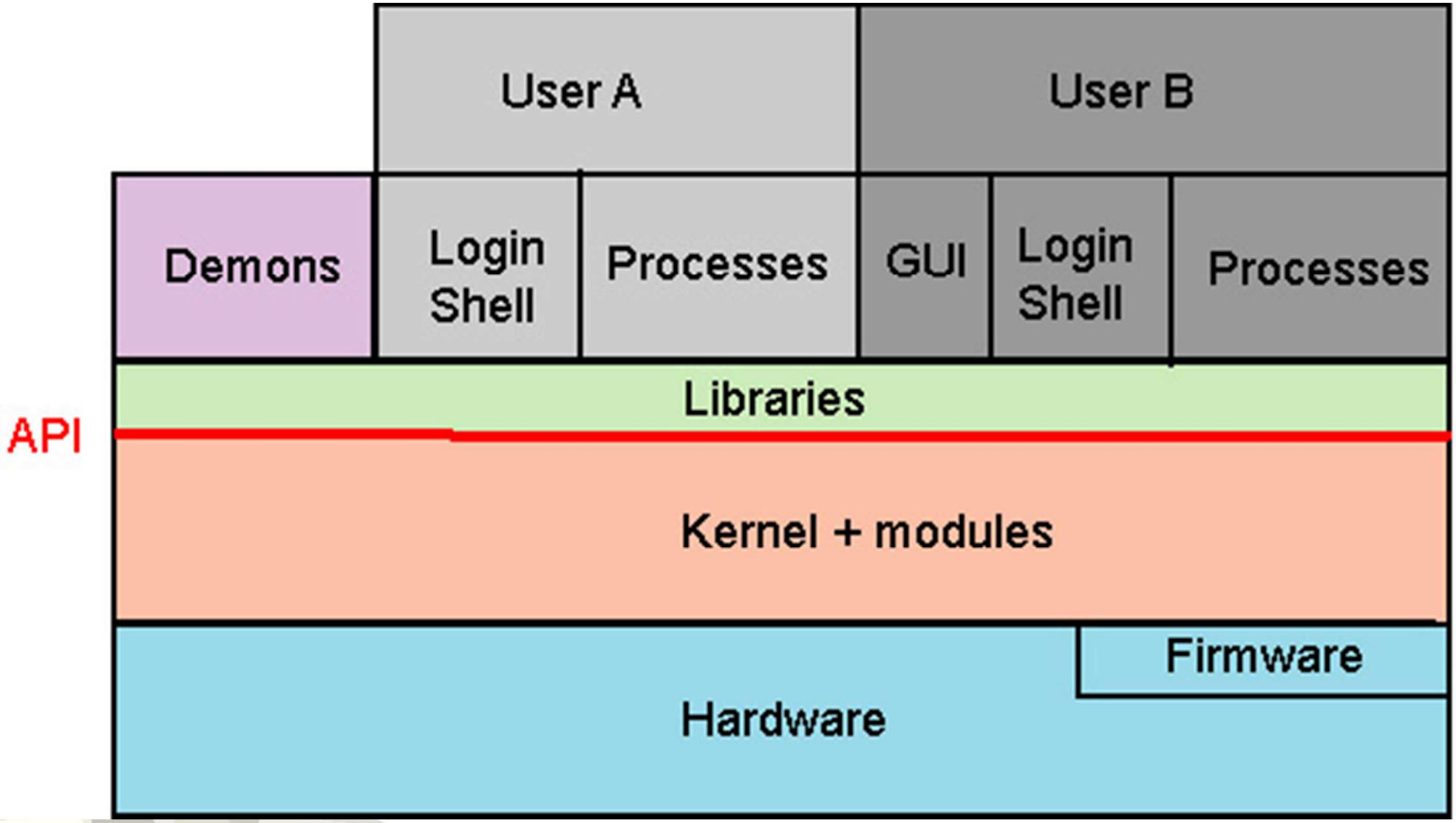
Unix: Structure, history and properties.

Shell and CLI.

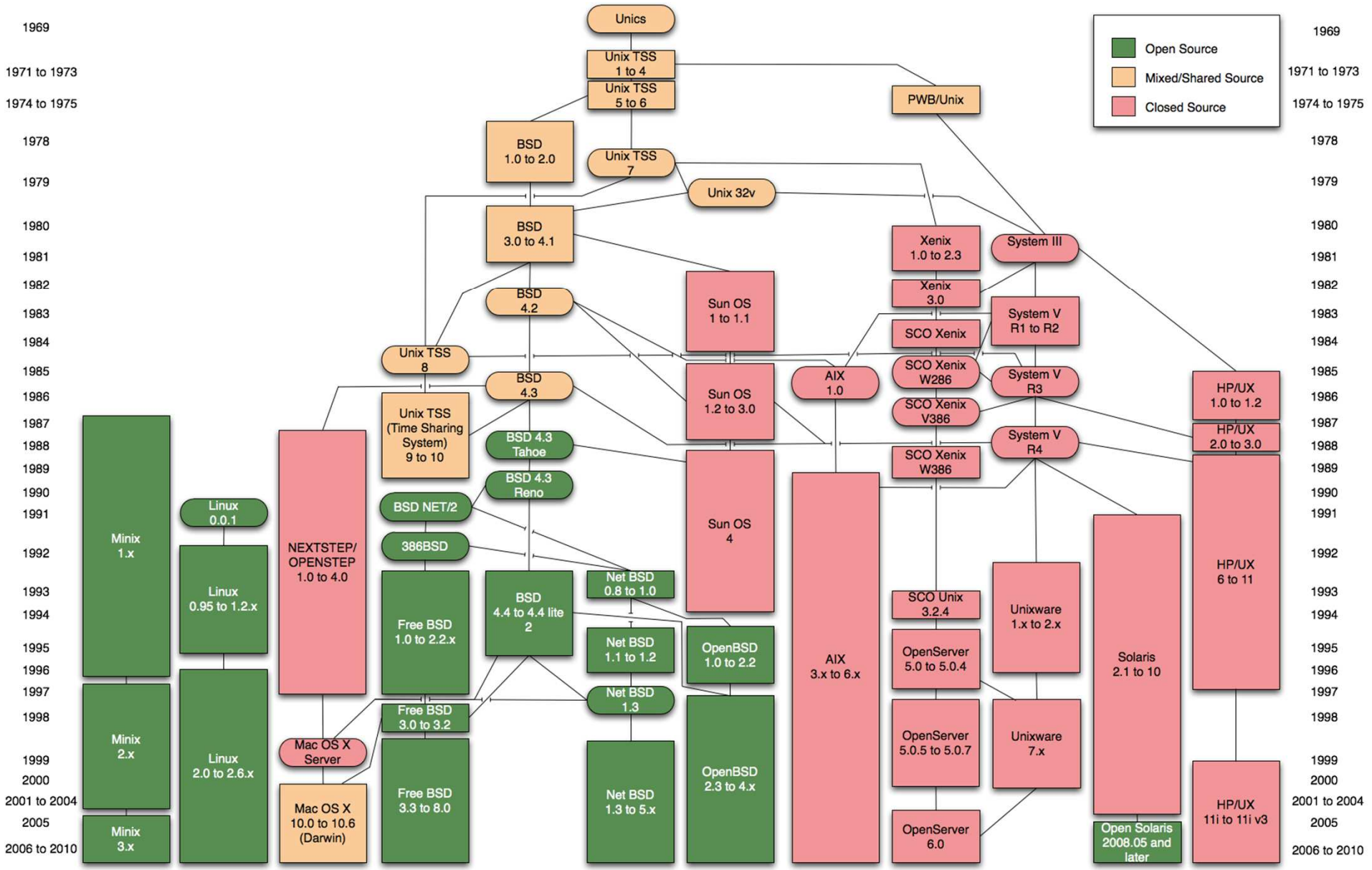


Department of Computer Systems FIT, Czech Technical University in Prague

Unix - structure



UNIX - history



Source: http://en.wikipedia.org/wiki/Unix

UNIX - properties

- **Portable**
  - 90% of kernel is written in C
- **Multi-user**
- **Multitasking, time-sharing**
- **Multithreading**
- **Symmetric Multi Processing (SMP)**
- **CLI**
- **IO redirection**
- **Hierarchical FS**
- **TCP/IP networking, NFS,…**
- **GUI**
  - X-Windows
  - Window managers - CDE, GNOME, KDE,…

Shell – command interpreter

- **CLI**
  - Command line parsing
  - Command execution
- **Shell scripts**
  - Script = Unix commands + control structures (e.g. loops, if/else…)
- **Environment**
  - Shell variables can define application behavior

Bourne shells

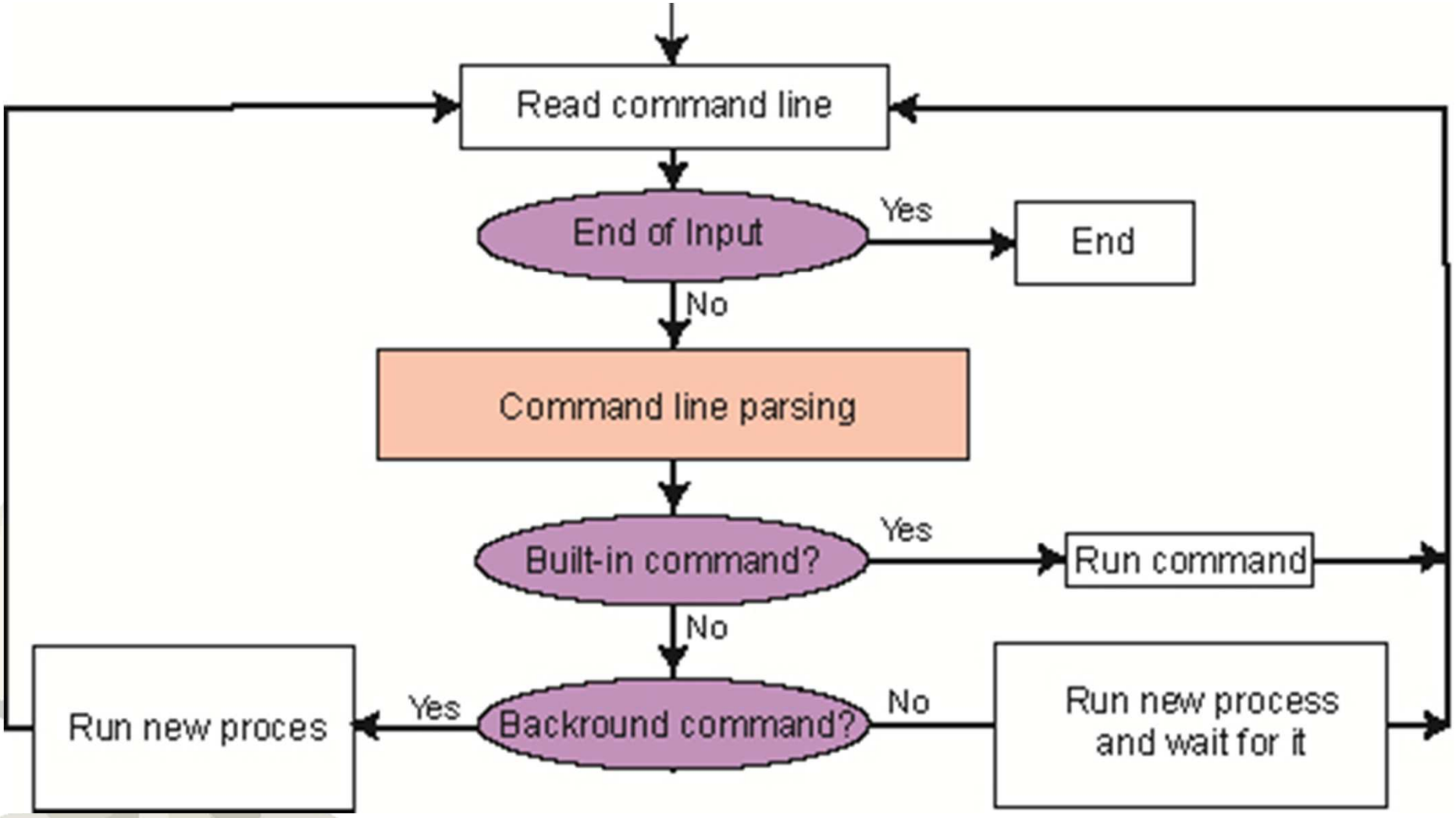
Name	File	Properties
Bourne shell	/bin/sh	basic
Korn shell	/bin/ksh	command history, job control, aliases,…
Bourne again shell	/bin/bash	like ksh but more user friendly

C shells

Name	File	Properties
C shell	/bin /csh	like ksh
Toronto C shell	/bin/tcsh	like csh, but more user friendly

- More information about shell we can find in Unix manual (e.g. man bash).
- In this modules we concentrate to Bourne shells.

Command line parsing



Command line syntax

- **Variables**

  `<prompt> <variable_name>=<value>`

  `<prompt>`

  prompt is printed by shell

  value of prompt is defined by shell variable PS1

  `<variable_name>`

  variable name is identifier

  shell set up the value to the variable

  `<value>`

  by default it is string

Command line syntax

Simple commands

```
<prompt> <command_name> <options> <arguments>
```

<command_name>

- define which program will be executed (which)
- it can be only name or path to the file (relative/absolute)

<options>

- can modify the behavior of command (how)

<arguments>

- specify the input date (what)

Command name, options and arguments are available

- in script by variables `$#`, `$0`, `$1`, `$2`, …
- in C program by variables `argc` , `argv[0]`, `arg[1]`, …

Examples

```
ls
ls /etc
ls -la /etc

B=`ypcat passwd | cut -d: -f1`
echo $B
echo "$B"

export LC_TIME=cs_CZ   ;   /usr/bin/echo "Dnes je \c"  ;   date '+%A %d.%m.%Y'

ypcat passwd | grep "student" | grep -v "docasne konto" | \
   sort -t':' -k3,3n| tail -1 | cut -d: -f 5 | cut -d' ' -f1,2
```

Is it clear??? Too simple???

Examples

Little bit more complicated?

```
echo PID FD EXEC FILENAME; PID=$(pgrep ''); pfiles $PID | awk 'BEGIN {
    fd=-1; }  /^[0-9]/ { if (fd>=0) { print pid, fd, exec; fd=-1; };
    pid=substr($1,0,length($1)-1); exec=$2; } /^ *[0-9]*: / { if (fd>=0) { print
    pid, fd, exec; fd=-1; }; if ($2=="S_IFREG") { fd=substr($1,0,length($1)-1); } }
    /^ *\// { fd=-1; }' | while read pid fd exec; do echo $pid $fd $exec $(echo
    0t$pid ::pid2proc \| ::fd $fd \| ::print file_t f_vnode \| ::vnode2path | mdb
    -k 2>/dev/null); done
```