Malloc(3) Revisited

Poul-Henning Kamp
UNIX wizard at large.

The FreeBSD Project

malloc(3) API

```
#include <stdlib.h>

void *malloc(size_t size)

void *calloc(size_t number, size_t size)

void *realloc(void *ptr, size_t size)

void free(void *ptr)

char *malloc_options;
```

Why Bother ?

We needed a better malloc for FreeBSD

We're not too happy about the GNU license

"GNU malloc isn't **that** great anyway"

RAM was very expensive at the time

What makes a malloc "good" ?

Efficiency

Error detection

Error handling

Debugging aids

Efficiency ?

Overhead

how long time does it spend doing what it does.

Quality of allocation

how well does it manage the RAM.

Spying with malloc(3)

new syscall:

```
void utrace __P((struct ut *, int));
```

Ask malloc to report to us:

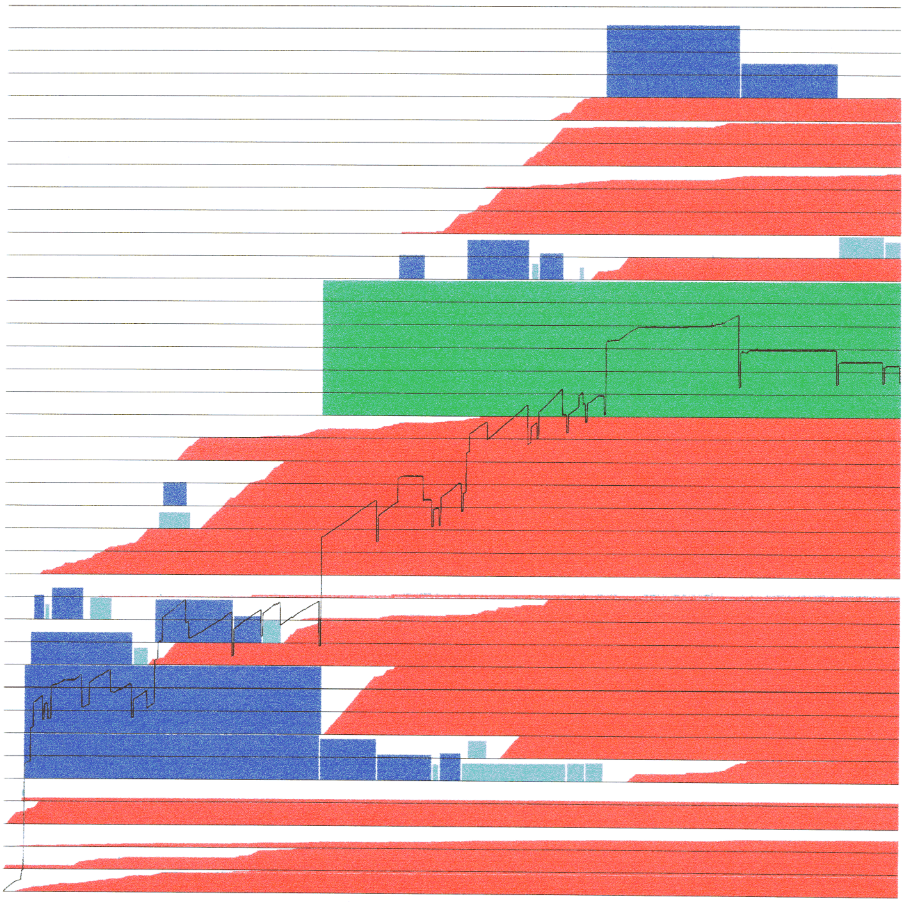
```
% setenv MALLOC_OPTIONS U
% ktrace -t u command_on_the_teststand
```

Spying with malloc(3) /2

```
% kdump

1619 a.out    USER  12  00 00 00 00 78 00 00 00 00 40 01 00

1619 a.out    USER  12  00 40 01 00 f0 00 00 00 00 50 01 00

1619 a.out    USER  12  00 50 01 00 00 00 00 00 00 00 00 00

                        ----------- ----------- -----------
                             a           b           c
```

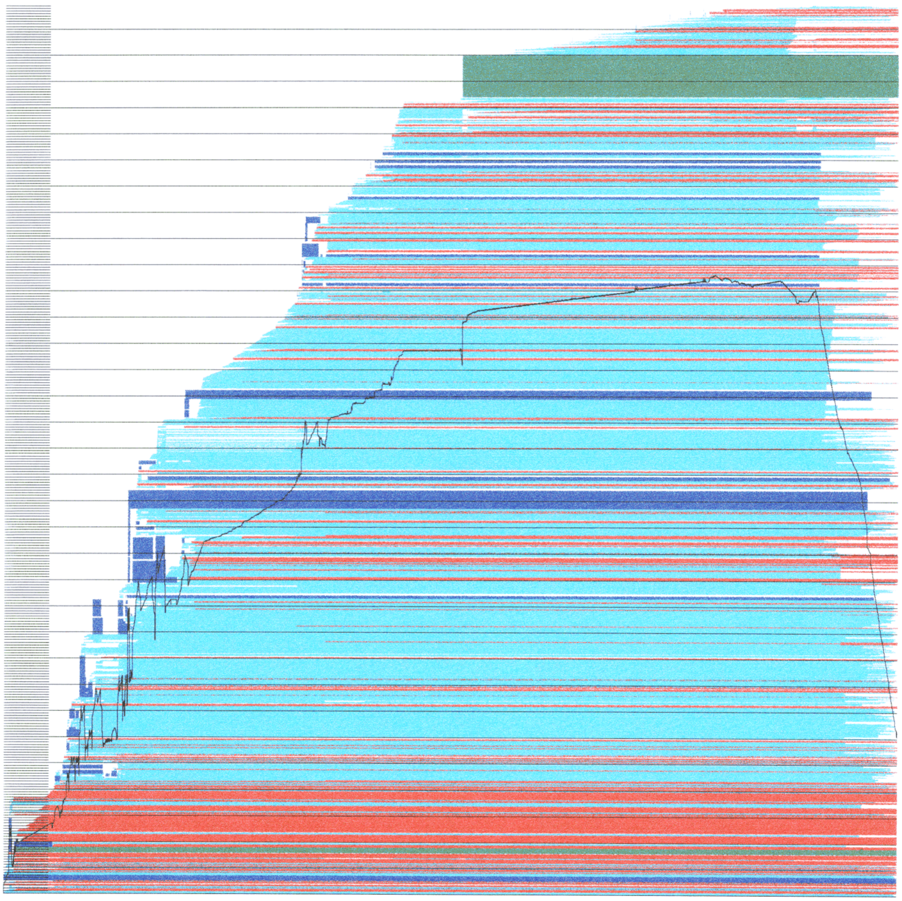
```
c = malloc(b);

free(a);

c = realloc(a, b);
```

Pid 1677: cpp

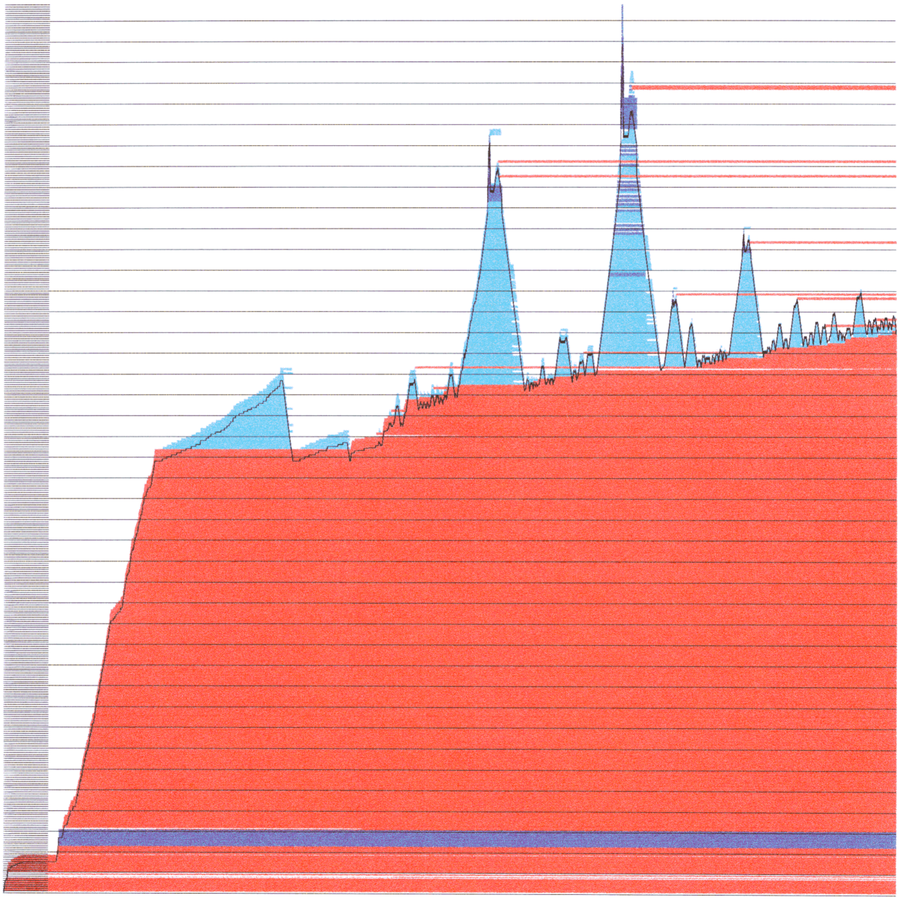


RED:	$\leq$PAGE	!freed
Cyan:	$\leq$PAGE	freed
green:	$\geq$PAGE	!freed
blue:	$\geq$PAGE	freed

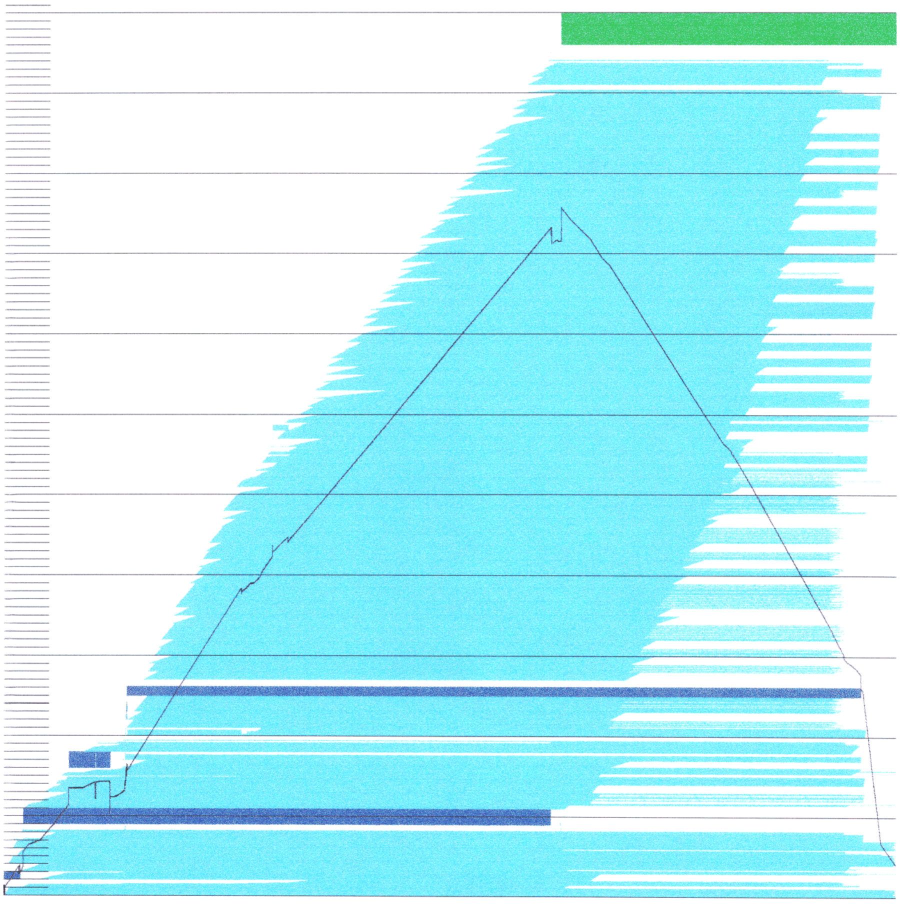
Pid 3463: wish8.0



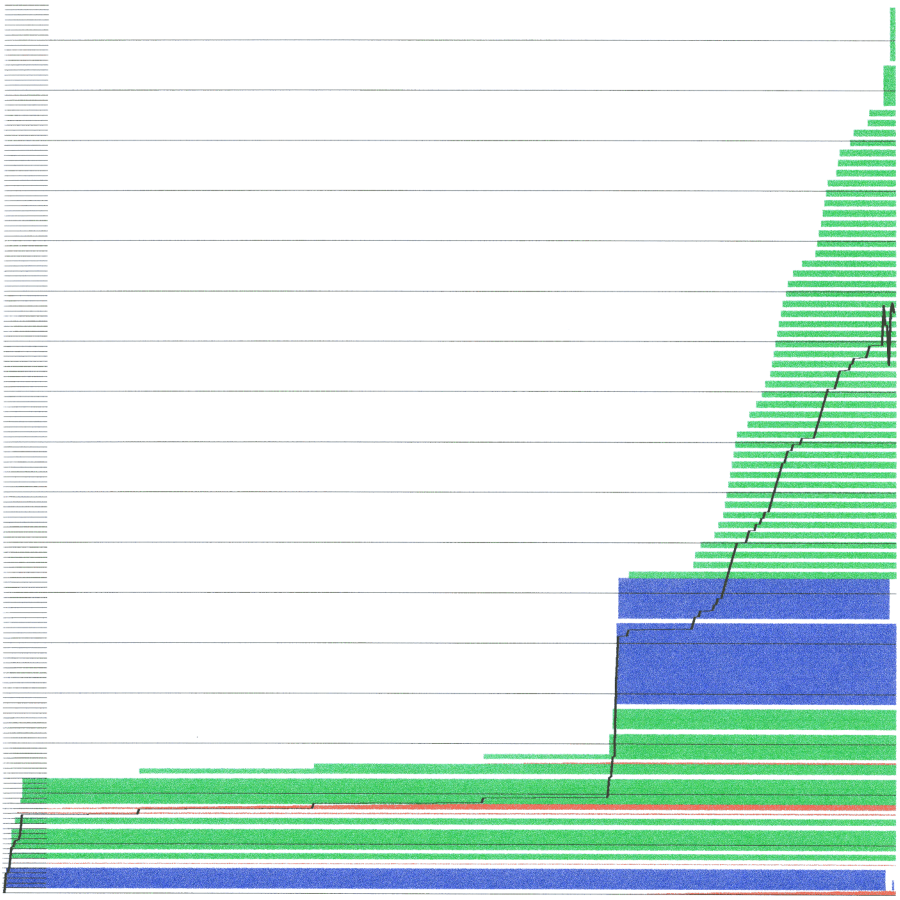
Pid 2055: cc1



Pid 2052: make



Pid 2056: as



Quality of allocation - then

"minimize size of program"

Either program is in RAM or on SWAP

Increased size increases chance of swapping.

Quality of allocation - Now

"minimize number of pages accessed."

Reduces paging rate.

Where malloc fail the test

Keeps free list structure in free mem

To access freelist, pages in unused pages

Up to 80% of time spent on paging otherwise unused pages.

Solution:

Don't store tiny data structures in huge pieces of free mem.

Error detection and all that...

Detect insane pointers:

too low (text, data, bss)

too high (mmap, shlibs, stack)

modified pointers

already free(3)'ed

Debugging and error detection

Junkfill allocations

Catches missing initialization

`0xd0` gives coredumps *[0xDuH!]*

Zerofill allocations

Verifies diagnosis

Workaround until fixed

Debugging/2

Abort on problems

Kill the process with corefile

Xmalloc

"But if we fail, We then can do't at land!"

Shakespeare

Passing options to malloc(3)

```
char *malloc_options = "x";
```

```
setenv MALLOC_OPTIONS AJ
```

```
ln -s AJ /etc/malloc.conf
```

A surprise in realloc()...

```
char *p = malloc(100);

p = realloc(p,0);

if(p)

        printf("Raise your hand");

else

        printf("Raise your hand");
```

The 'V' option...

Did it make a difference ?

fsck
ypserv
cvs
libkvm
libc:getpwent.c
libc:getvfsent.c
ypxfr
libproplist (whatever that is)
xcept (do)
ucd-snmp
mountd
symorder
ranlib
join
rpc.yppasswdd
inetd
crunchgen.c
amd
ppp

Weird idea department:

Use file in $HOME for backing (Quota!)

Don't free until we have run for 10 seconds

"I'm transient - Don't bother"

SIGVM - Change in VM status:
green: Don't worry
yellow: Please free as convenient
red: Free everything!

Availability:

```
/*
 * ----------------------------------------------------------------------------
 * "THE BEER-WARE LICENSE" (Revision 42):
 * <phk@FreeBSD.org> wrote this file.  As long as you retain this notice you
 * can do whatever you want with this stuff. If we meet some day, and you think
 * this stuff is worth it, you can buy me a beer in return.   Poul-Henning Kamp
 * ----------------------------------------------------------------------------
 */
```

Pick it up from FreeBSD

`http://www.freebsd.org/cgi/cvsweb.cgi/src/lib/libc/stdlib/malloc.c`

Questions ?