Linking Disk Schedulers into Xenix
(or, how I learned to stop worrying and love the kernel)

1. Introduction.

The goal of this project is to incorporate your own disk scheduling algorithm into the Xenix operating system. Three machines will be used, *moe* a production machine, and *larry* and *curly*, the crash-test machines. All of them are Intel 310/286's running Xenix 3.5. Accounts will be given out on *moe* only, work on the crash machines will be done as root. When you are ready to test your scheduler, you will transfer it to *moe* (if it isn't there already). From there you will transfer it to a crash-test machine, link it into the kernel, and then boot it. Running through this linking process with the attached example program is recommended.

2. Writing Disksort.

Under Xenix the disk scheduler is called *disksort*, its function is to enqueue disk I/O requests. A skeleton format for this program is:

```
/*
 *  XENIX include files
 */
#include "/spam/disksort/h/param.h"
#include "/spam/disksort/h/buf.h"
#include "/spam/disksort/h/conf.h"
#include "/spam/disksort/h/dir.h"
#include "/spam/disksort/h/a.out.h"
#include "/spam/disksort/h/user.h"
#include "/spam/disksort/h/systm.h"
#include "/spam/disksort/h/iobuf.h"


disksort(head, request)
struct iobuf        *head;              /* the header */
struct buf *request;            /* the request to enqueue */
{
        .
        .
        .
        .
}
```

The example program shows these *include*'s and the structure of the request queue. The example program is on *moe* in /spam/disksort/ds.c.

Disksort is called by every disk device-driver, therefore, no hardware specifics are known within *disksort*.

The first node in the queue is the *active* request, this means it is currently being referenced through by the device. If a request is inserted before the *active* request this new request will be referenced through, effectively scrambling the disk blocks.

3. Transferring Files

You can prepare your disksort program on any system, but to run it, you will have to transfer it to *moe*. *Moe* can transfer files from any machine you can get to via malach (the terminal server) or it can directly read ms-dos disks.

To get a file from *jove* or *psu-cs* you will use the *cu* (call unix) command. Cu allows you to call out through a port to connect with another system. Cu can download and upload files from/to the remote host. During your session you can ask *cu* to do various things, all these commands are prefixed with a return-tilde (˜). The following is a sample session in which the file on psu-cs called 'spam' is copied into the current directory (from now on, '%' represents the prompt):

% cu -lttyf0 dir	*connect to ttyf0*
connected	*hit return*
:	
:	
malach> rlogin jove	*login to your favorite account*
:	
:	
%	*make sure to hit return before the '˜'*
˜%take spam	*...and after the command*
:	
11 lines/384 characters	
% logout	*make sure to logout before disconnecting*
malach>	
˜.	*hit return-tilde-dot-return*
disconected	

Note that the session is terminated with return-tilde-dot-return.

Xenix provides the following commands for working with ms-dos disks: doscat, doscp, dosdir, and dosls. The following is an example session, in which the ms-dos file 'spam' is copied into the current directory:

% dosls A:	*A: specifies the floppy drive*
head.exe	
spam	
% doscp A:spam .	
% ls	
spam	

For more information on either of these see the Xenix manuals in the Tutor's room.

4. Transfer to the crash machine

Editing of your disk scheduler can be done on any system, but to test it you will have to transfer it to *moe* When your program is ready to be tested, place a floppy disk (a few will be floating around in the terminal room) into *moe*'s floppy drive and type

% makeds *file*

where *file* is the name of your disk scheduler (with a '.c' suffix). This will copy your disk scheduler onto the floppy disk. When *makeds* is done, take the floppy disk out and place it in either *larry* or *curly*'s floppy drive.

5. Making a Kernel.

To transfer your disk scheduler from the floppy and link it into Xenix, log in as root on the crash-test machine and type:

% mkboot

This will create a new kernel, called *xenix.dsa,* in the root directory. After doing this you are ready to boot your modified kernel.

6. Booting your Kernel.

First the crash machine has to be shut down, do this by typing (you have to be logged in as root to do this):

```
% sync
% sync
% sync
% haltsys
```

You should then see the message:

```
*** Normal System Shutdown ***
```

The system is now ready to be rebooted. On the front of the computer at the lower left, near the base, are a pair of square buttons labeled $\boxed{R}$ (reset) and $\boxed{I}$ (interrupt). To begin the rebooting process hit $\boxed{R}$ followed by $\boxed{I}$. and then type U (shift u) within a few seconds, you should then be greeted with:

```
iSDM 286 Monitor V1.0
Copyright 1983 Intel Corporation
.
```

The period you see is the prompt for the machine level monitor, at this point you can direct it to boot your modified kernel by typing:

```
b xenix.dsa
```

The system will then run through the boot process, and then say:

```
Type Ctrl-D to proceed with normal startup,
(or give root password for system maintenance):
```

Hit Ctrl-D to bring the system up multi-user. It will then request the date and time, (just hit return) and it will then give a login prompt. Your kernel is running!

7. Benchmarks

The benchmark program can be run by logging in and typing bench on a crash machine. This program prints the starting time and starts up several processes, one of them is called *cpuhog*, it does no I/O but takes up most of the free CPU time on the system. The other processes do several I/O intensive operations, when these are done, the *cpuhog* is sent a signal to stop and print the number of iterations. Then the finishing time is printed.

If you wish to look at the benchmark programs, they are in /usr/local.

8. If something goes wrong.

If after booting using your kernel, the system locks up, or starts issuing errors, etc... it can be stopped by hitting $\boxed{R}$ followed by $\boxed{I}$ (shutting the machine off should be saved as a last resort). This will restart the boot process, go through the aforementioned boot procedure, except at the period prompt just type b with no arguments. This will boot the standard kernel (*xenix*).

9. If you need help.

Trent Fisher will be available to assist you on Tuesday and Thursday from 10:30 to 13:00, and Saturday from 14:00 to 18:00. He may also be contacted via E-mail: trent@psu-cs.