CMPS 240 – Sprint 3
Team: *Cachers*
Members: *Abdul Rahman Kobeissi*
Stephane Najjar
Yara Khalifeh

LINK TO GITHUB REPOSITORY:
https://github.com/yrk04/CMPS240-xv6-project

OBJECTIVE:

MODIFICATIONS:

FILES	
fs.h	- Defined MAX_EXTENTS the max number of extents supported by a file. - Defined the extents struct, storing two uints: start and length to denote the start and length of our extent of blocks - Modified the dinode struct to include an array of extents. - Included char padding[16] due to an error occurring because the size of extents did not divide the size of blocks
file.h	- Included “fs.h” so that file.h had access to the extents struct. - Added an extent array to the inode struct
stat.h	- Defined a 4th file type, T_EXTENT 4 - Included case T_EXTENT: in the switch(st.type) block to show "extent file" in the output of stat.
ls.c	- Inserted a case T_EXTENT: in the switch(st.type) block to print "extent" when listing files.
fs.c	Changes have been made to several methods:
readi	- New section that executes if file is of

	type T_EXTENT: - calculate the block number that the offset would read to - try to find it in the extents of the file - if found, read the data - else, move on to the next extent until found or until done with extents. - return total number of bytes read
writei	- Much like readi, added a new section that executes if file is of type T_EXTENT - calculate block number to write to - try to find it in the extents - if found, write to it - else, move on to the next extent until found or done with extents - update file size if needed - return total number of bytes written
itrunc	- If file is of type T_EXTENT - free every block - clear the extents in the inode (set all their values to zero) - update file size to 0.
ialloc	- If file is of type T_EXTENT - initialize extents to 0
ilock	- If file is of type T_EXTENT - load the extent data into the inode from the dinode.
sysproc.c	Added new system call implementation: `int sys_getextents(void)` to print extents from inode
defs.h	Declared: `int sys_getextents(void);` so the syscall is accessible in other kernel files
syscall.c	Added to syscall dispatch table: `[SYS_getextents] sys_getextents,`
syscall.h	Defined syscall number: `#define SYS_getextents 24` *(or next available number)*
user.h	Declared user-facing syscall: `int getextents(const char *);`
usys.S	Added syscall entry: `SYSCALL(getextents)` to map user call to kernel handler

extenttest.c	Created test program: 1. Opens/creates `myfile.txt` 2. Writes 10 blocks of data 3. Calls `getextents("myfile.txt")` to print extents from the kernel
Makefile	Added `_extenttest\` to the `UPROGS` list

sidenote: a bunch of header files were updated because they (unsure why) started causing redefinition errors. added #ifndef, #define and #endif.