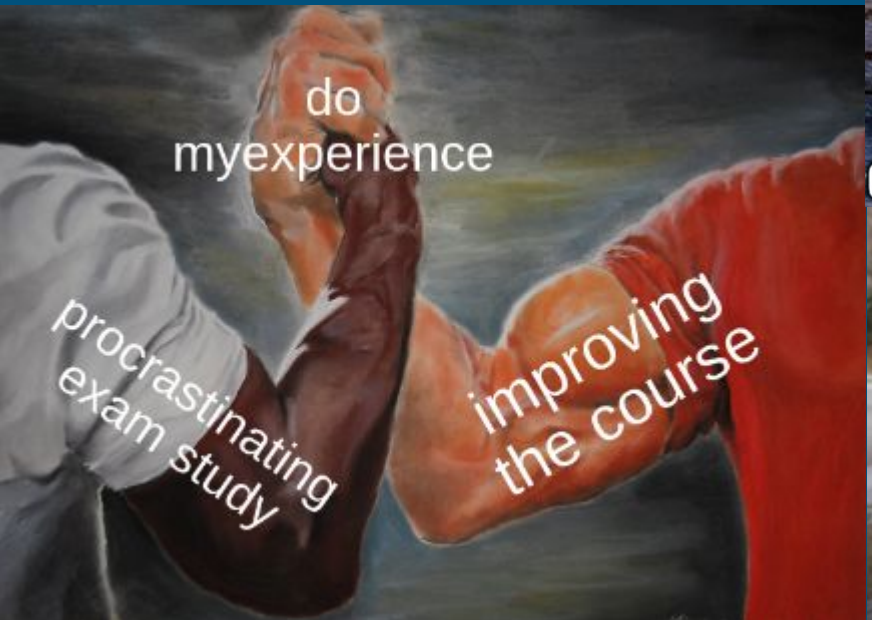
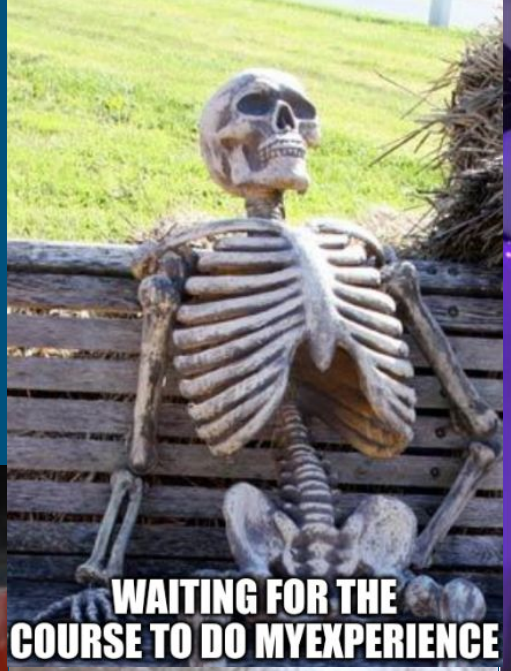
COMP1521 Week 9

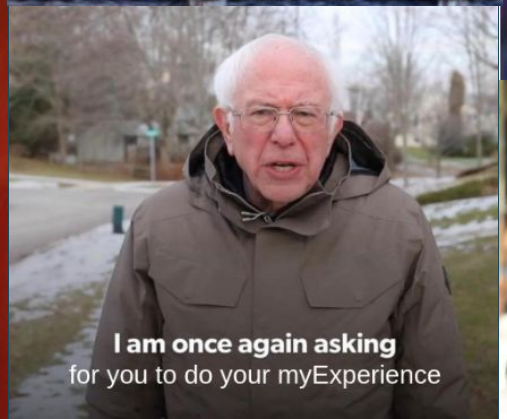
(Week 2 of) Intro to Operating Systems

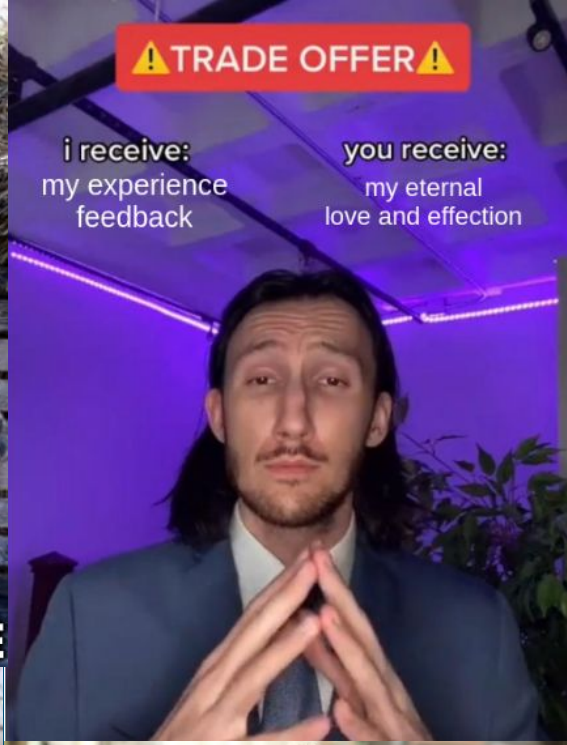
MyExperience

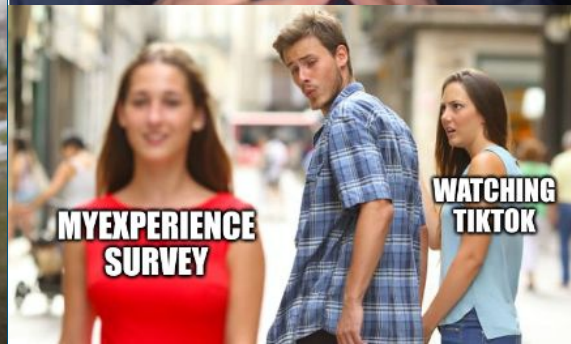
- Please do it











What are environment variables?

- A way for processes to gain information about the environment that they are running in.
- Examples include $HOME (home directory), $PWD (current working directly), and $OLD_PWD (previous directory - this is how “cd -” can take you to the previous directory.
- Let’s see an example in code!

The stat() and lstat() functions both take an argument which is a pointer to a struct stat object, and fill it with the meta-data for a named file.

On Linux, a struct stat contains the following fields (among others, which have omitted for simplicity):

```
struct stat {
    ino_t st_ino;         /* inode number */
    mode_t st_mode;       /* protection */
    uid_t st_uid;         /* user ID of owner */
    gid_t st_gid;         /* group ID of owner */
    off_t st_size;        /* total size, in bytes */
    blksize_t st_blksize; /* blocksize for filesystem I/O */
    blkcnt_t st_blocks;   /* number of 512B blocks allocated */
    time_t st_atime;      /* time of last access */
    time_t st_mtime;      /* time of last modification */
    time_t st_ctime;      /* time of last status change */
};
```

What is st_ino?

```
struct stat {
    ino_t st_ino;         /* inode number */
    mode_t st_mode;       /* protection */
    uid_t st_uid;         /* user ID of owner */
    gid_t st_gid;         /* group ID of owner */
    off_t st_size;        /* total size, in bytes */
    blksize_t st_blksize; /* blocksize for filesystem I/O */
    blkcnt_t st_blocks;   /* number of 512B blocks allocated */
    time_t st_atime;      /* time of last access */
    time_t st_mtime;      /* time of last modification */
    time_t st_ctime;      /* time of last status change */
};
```

The inode number! Each file has an inode somewhere on disk that stores metadata about the file (location on disk, file size, etc.). Inode number can be accessed using ls -li

What is st_mode?

```
struct stat {
    ino_t st_ino;          /* inode number */
    mode_t st_mode;        /* protection */
    uid_t st_uid;          /* user ID of owner */
    gid_t st_gid;          /* group ID of owner */
    off_t st_size;         /* total size, in bytes */
    blksize_t st_blksize; /* blocksize for filesystem I/O */
    blkcnt_t st_blocks;    /* number of 512B blocks allocated */
    time_t st_atime;       /* time of last access */
    time_t st_mtime;       /* time of last modification */
    time_t st_ctime;       /* time of last status change */
};
```

We can bitwise and (&) st_mode with S_IRUSR to check if a user can read, or S_IWGRP to see if a group can write to it. You can pass it to macros like S_ISREG(m) to check if it is a regular file.

A bit field that represents the permissions for the file. These are generally treated as octal numbers, to make the 3 bits defining permissions for users/groups/global easy to read.

What is st_uid?

```
struct stat {
    ino_t st_ino;         /* inode number */
    mode_t st_mode;       /* protection */
    uid_t st_uid;         /* user ID of owner */
    gid_t st_gid;         /* group ID of owner */
    off_t st_size;        /* total size, in bytes */
    blksize_t st_blksize; /* blocksize for filesystem I/O */
    blkcnt_t st_blocks;   /* number of 512B blocks allocated */
    time_t st_atime;      /* time of last access */
    time_t st_mtime;      /* time of last modification */
    time_t st_ctime;      /* time of last status change */
};
```

Gives user id of the file’s owner. Can be accessed with ls -ln

What is st_gid?

```
struct stat {
    ino_t st_ino;         /* inode number */
    mode_t st_mode;       /* protection */
    uid_t st_uid;         /* user ID of owner */
    gid_t st_gid;         /* group ID of owner */
    off_t st_size;        /* total size, in bytes */
    blksize_t st_blksize; /* blocksize for filesystem I/O */
    blkcnt_t st_blocks;   /* number of 512B blocks allocated */
    time_t st_atime;      /* time of last access */
    time_t st_mtime;      /* time of last modification */
    time_t st_ctime;      /* time of last status change */
};
```

Gives group id of the file’s owner. Can be accessed with ls -ln

What is st_size?

```
struct stat {
    ino_t st_ino;         /* inode number */
    mode_t st_mode;       /* protection */
    uid_t st_uid;         /* user ID of owner */
    gid_t st_gid;         /* group ID of owner */
    off_t st_size;        /* total size, in bytes */
    blksize_t st_blksize; /* blocksize for filesystem I/O */
    blkcnt_t st_blocks;   /* number of 512B blocks allocated */
    time_t st_atime;      /* time of last access */
    time_t st_mtime;      /* time of last modification */
    time_t st_ctime;      /* time of last status change */
};
```

Total file size in bytes. For simple text files, this will simply be the length of the content.

What is st_blksize?

```
struct stat {
    ino_t st_ino;         /* inode number */
    mode_t st_mode;       /* protection */
    uid_t st_uid;         /* user ID of owner */
    gid_t st_gid;         /* group ID of owner */
    off_t st_size;        /* total size, in bytes */
    blksize_t st_blksize; /* blocksize for filesystem I/O */
    blkcnt_t st_blocks;   /* number of 512B blocks allocated */
    time_t st_atime;      /* time of last access */
    time_t st_mtime;      /* time of last modification */
    time_t st_ctime;      /* time of last status change */
};
```

Files are divided into chunks (blocks) on disk. st_blksize gives the size of a block on the storage device generally used for files of this type. Typical block sizes are 512, 1024, 4096, 8192.

What is st_blocks?

```
struct stat {
    ino_t st_ino;         /* inode number */
    mode_t st_mode;       /* protection */
    uid_t st_uid;         /* user ID of owner */
    gid_t st_gid;         /* group ID of owner */
    off_t st_size;        /* total size, in bytes */
    blksize_t st_blksize; /* blocksize for filesystem I/O */
    blkcnt_t st_blocks;   /* number of 512B blocks allocated */
    time_t st_atime;      /* time of last access */
    time_t st_mtime;      /* time of last modification */
    time_t st_ctime;      /* time of last status change */
};
```

Shows the number of 512 byte blocks allocated for the file. Often these will be allocated in groups of 2^n blocks, though, so a file that is only 1855 bytes may be allocated 8 blocks (or 4096 bytes) for example.

What is st_atime?

```
struct stat {
    ino_t st_ino;         /* inode number */
    mode_t st_mode;       /* protection */
    uid_t st_uid;         /* user ID of owner */
    gid_t st_gid;         /* group ID of owner */
    off_t st_size;        /* total size, in bytes */
    blksize_t st_blksize; /* blocksize for filesystem I/O */
    blkcnt_t st_blocks;   /* number of 512B blocks allocated */
    time_t st_atime;      /* time of last access */
    time_t st_mtime;      /* time of last modification */
    time_t st_ctime;      /* time of last status change */
};
```

Gives the last time the file was modified. A time_t value is typically implemented as an integer giving the number of seconds since midnight on Jan 1 1970.

What is st_mtime?

```
struct stat {
    ino_t st_ino;         /* inode number */
    mode_t st_mode;       /* protection */
    uid_t st_uid;         /* user ID of owner */
    gid_t st_gid;         /* group ID of owner */
    off_t st_size;        /* total size, in bytes */
    blksize_t st_blksize; /* blocksize for filesystem I/O */
    blkcnt_t st_blocks;   /* number of 512B blocks allocated */
    time_t st_atime;      /* time of last access */
    time_t st_mtime;      /* time of last modification */
    time_t st_ctime;      /* time of last status change */
};
```

The last time the file content was accessed (read or written). Can be accessed using ls -lu.

What is st_ctime?

```
struct stat {
    ino_t st_ino;         /* inode number */
    mode_t st_mode;       /* protection */
    uid_t st_uid;         /* user ID of owner */
    gid_t st_gid;         /* group ID of owner */
    off_t st_size;        /* total size, in bytes */
    blksize_t st_blksize; /* blocksize for filesystem I/O */
    blkcnt_t st_blocks;   /* number of 512B blocks allocated */
    time_t st_atime;      /* time of last access */
    time_t st_mtime;      /* time of last modification */
    time_t st_ctime;      /* time of last status change */
};
```

The last time the file status was changed. This means either changing the contents or the metadata. This value can be accessed using ls -lc.

This output is from the command “ls -l ~cs1521”

```
drwxr-x--- 11 cs1521 cs1521 4096 Aug 27 11:59 17s2.work
drwxr-xr-x  2 cs1521 cs1521 4096 Aug 20 13:20 bin
-rw-r-----  1 cs1521 cs1521   38 Jul 20 14:28 give.spec
drwxr-xr-x  3 cs1521 cs1521 4096 Aug 20 13:20 lib
drwxr-x--x  3 cs1521 cs1521 4096 Jul 20 10:58 public_html
drwxr-xr-x 12 cs1521 cs1521 4096 Aug 13 17:31 spim
drwxr-x---  2 cs1521 cs1521 4096 Sep  4 15:18 tmp
lrwxrwxrwx  1 cs1521 cs1521   11 Jul 16 18:33 web -> public_html
```

Who can access 17s2.work? What type of file is it?

```
drwxr-x--- 11 cs1521 cs1521 4096 Aug 27 11:59 17s2.work
drwxr-xr-x  2 cs1521 cs1521 4096 Aug 20 13:20 bin
-rw-r-----  1 cs1521 cs1521   38 Jul 20 14:28 give.spec
drwxr-xr-x  3 cs1521 cs1521 4096 Aug 20 13:20 lib
drwxr-x--x  3 cs1521 cs1521 4096 Jul 20 10:58 public_html
drwxr-xr-x 12 cs1521 cs1521 4096 Aug 13 17:31 spim
drwxr-x---  2 cs1521 cs1521 4096 Sep  4 15:18 tmp
lrwxrwxrwx  1 cs1521 cs1521   11 Jul 16 18:33 web -> public_html
```

- It's a directory
- The user cs1521, and the group cs1521 can cd into it and list the files. The cs1521 user can also write to the directory.

What operations can a typical user perform on the public_html directory?

```
drwxr-x--- 11 cs1521 cs1521 4096 Aug 27 11:59 17s2.work
drwxr-xr-x  2 cs1521 cs1521 4096 Aug 20 13:20 bin
-rw-r-----  1 cs1521 cs1521   38 Jul 20 14:28 give.spec
drwxr-xr-x  3 cs1521 cs1521 4096 Aug 20 13:20 lib
drwxr-x--x  3 cs1521 cs1521 4096 Jul 20 10:58 public_html
drwxr-xr-x 12 cs1521 cs1521 4096 Aug 13 17:31 spim
drwxr-x---  2 cs1521 cs1521 4096 Sep  4 15:18 tmp
lrwxrwxrwx  1 cs1521 cs1521   11 Jul 16 18:33 web -> public_html
```

- They can cd into it, but they can't list the directory contents or write to it. They could also, for example, read a file if they both knew the filename and that file had global read permissions.

What is the file web?

```
drwxr-x--- 11 cs1521 cs1521 4096 Aug 27 11:59 17s2.work
drwxr-xr-x  2 cs1521 cs1521 4096 Aug 20 13:20 bin
-rw-r-----  1 cs1521 cs1521   38 Jul 20 14:28 give.spec
drwxr-xr-x  3 cs1521 cs1521 4096 Aug 20 13:20 lib
drwxr-x--x  3 cs1521 cs1521 4096 Jul 20 10:58 public_html
drwxr-xr-x 12 cs1521 cs1521 4096 Aug 13 17:31 spim
drwxr-x---  2 cs1521 cs1521 4096 Sep  4 15:18 tmp
lrwxrwxrwx  1 cs1521 cs1521   11 Jul 16 18:33 web -> public_html
```

- A symbolic link (symlink) to public_html. It basically makes web another name for public_html.

What is the difference between stat("web", &info) and lstat("web", &info)? (where info is an object of type (struct stat))

```
drwxr-x--- 11 cs1521 cs1521 4096 Aug 27 11:59 17s2.work
drwxr-xr-x  2 cs1521 cs1521 4096 Aug 20 13:20 bin
-rw-r-----  1 cs1521 cs1521   38 Jul 20 14:28 give.spec
drwxr-xr-x  3 cs1521 cs1521 4096 Aug 20 13:20 lib
drwxr-x--x  3 cs1521 cs1521 4096 Jul 20 10:58 public_html
drwxr-xr-x 12 cs1521 cs1521 4096 Aug 13 17:31 spim
drwxr-x---  2 cs1521 cs1521 4096 Sep  4 15:18 tmp
lrwxrwxrwx  1 cs1521 cs1521   11 Jul 16 18:33 web -> public_html
```

- The stat() function call will follow the symlink to public_html, and place that metadata into the info struct.
- The lstat() function call places metadata about the symlink itself into the info struct.

Let’s see stat in practice!