Week 10 - Friday

COMP 2400

Last time

- What did we talk about last time?
- Finished time
- File I/O

Questions?

Project 5

Quotes

The first 90% of the code accounts for the first 90% of the development time. The remaining 10% of the code accounts for the other 90% of the development time.

Tom Cargill

Exam 2 Post Mortem

A Little More on File I/O

Error handling

- Lots of errors can happen with file I/O
- If a file cannot be opened with the given mode, **`fopen()`** returns **`NULL`** and **`errno`** is set to an appropriate error code
- The **`fprintf()`** function returns the number of characters written
  - A value less than or equal to 0 indicates error
- The **`fscanf()`** function returns the number of items read
  - If that number is less than expected, it's an error

Standard streams

- C programs that run on the command line have the following file pointers open by default
  - **`stdin`**
  - **`stdout`**
  - **`stderr`**
- You can use them where you would use other file pointers

Aliases for other functions

- You can think of the input and output functions you've been using as special cases of these file operations
  - They are often implemented that way
- For example:
  - **`getchar()`** is equivalent to **`fgetc(stdin)`**
  - **`printf(…)`** is equivalent to **`fprintf(stdout,…)`**
  - **`scanf(…)`** is equivalent to **`fscanf(stdin, …)`**

Users and Groups

Users

- Recall that each user on a Linux system has a unique login name and a unique numerical identifier (the UID)
- Users can belong to one or more groups as well
- Where is this information stored?

Password file

- The system has a password file stored in **`/etc/passwd`**
- Each line of this file corresponds to one user in the system and has seven fields separated by colons:
  - Login name
  - Encrypted password
  - UID
  - GID (group ID of the first group that the user is a member of)
  - Comment
  - Home directory (where you are when you log in)
  - Login shell (which shell you running when you log in)
- Example:

```
wittman1:x:1000:100:Barry Wittman:/home/wittman1:/bin/bash
```

Catch-22

- Your computer needs to be able read the password file to check passwords
- But, even **`root`** shouldn't be able to read everyone's passwords
- Hash functions to the rescue!

Cryptographic hash functions

- Takes a message of any size and turns it into a short, fixed-size digest
- Different from hash functions used for hash tables
- Lots of interesting properties (lots more than these):

Avalanching	• A small change in the message should make a big change in the digest
Preimage Resistance	• Given a digest, should be hard to find a message that would produce it
Collision Resistance	• Should be hard to find two messages that hash to the same digest (collision)

The Linux and Unix solution

- Instead of storing actual passwords, Linux machines store the hash of the passwords
- When someone logs on, the operating system hashes the password and compares it to the stored version
- No one gets to see your original password
  - Not even **`root`**!

Back to the password file

- Inside the password file, we have encrypted passwords
- Everyone's password is safe after all

Login Name	Password Hash
ahmad	IfW{6Soo
baili	853aE90f
carmen	D390&063
deepak	CWc^Q3Ge
erica	e[6s_N*X1

Shadow password file

- Even though the password is disguised, it's unwise to leave it visible to everyone
  - Given a password digest (the hashed version) and lots of time, it is possible to figure out the password
- It's useful for the password file to be readable by everyone so that all users on a machine are known to all others
- A shadow password file stores the encrypted password and is readable only by privileged users
  - **`/etc/shadow`**

Changing your password

- Amid all this discussion, it might be useful to know how to change your password
- I **don't** recommend that you do change your password
  - I'm honestly not sure how doing so will interact with your Active Directory (Windows) password
- The command is **`passwd`**

```
Changing password for wittman1.
(current) UNIX password:
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
```

Changing the owner of a file

- You recall that we can change permissions for who can read, write, and execute a file using **`chmod`**
- But **`chmod`** depends on who the owner is
- What if you want someone else to be the owner of a file?
- The **`chown`** command can let you do that
- If I want my file **`stuff.txt`** to be owned by Professor Stucki, I would use the following command

```
chown dstucki stuff.txt
```

- On most systems, **`chown`** only works if you are **`root`**

Groups

- Files are associated with a group as well as a user who is owner
- The groups are listed in the **`/etc/group`** file
- Each line of this file corresponds to a group and has four fields separated by colons:
  - Group name
  - Encrypted password
    - Often not used
  - Group ID (GID)
  - User list
    - Comma separated

```
users:x:100:
jambit:x:106:claus,felli,frank,harti,markus,martin,mtk,paul
```

Creating a group

- If you want to create a group, you have to be **root**
- If you're **root** (or using **sudo**), you can use the **groupadd** command
- To create the **awesome** group as **root**:

```
groupadd awesome
```

- Or using **sudo**:

```
sudo groupadd awesome
```

Adding a user to a group

- Again, you have to be **root** to add a user to a group
- Use the **useradd** command
- To add user **wittman1** to the **awesome** group as **root**:

```
useradd -g awesome wittman1
```

- Or using **sudo**:

```
sudo useradd -g awesome wittman1
```

Changing the group for a file

- When you create a file, it is associated with some default group that you belong to
- You can use the **chgrp** command to change to another group that you belong to

```
chgrp awesome file.txt
```

- If you are root, you can use the **chown** command to change the group, using a colon

```
chown :awesome file.txt
```

Upcoming

Next time…

- More on binary files
- Low-level I/O

Reminders

- Work on Project 5
- Read LPI chapters 13, 14, and 15