Portland State
Computer Science

CS 333

mproc_crypt

Portland State
Computer Science

How we got here…

While working with the **`crypt()`** functions, I thought about what would happen if an especially long running hashed password was passed to **`crypt()`**. I'd like for **`crypt()`** to time out after some period. I could just skip that hashed password and move on to the next one.

Unfortunately, the **`crypt()`** functions do have a timeout parameter.

My initial through was to have an alarm (using the **`alarm()`** function) go off that would generate a signal (**`SIGALRM`**). Many system functions fail if a signal is delivered during a call. I could handle the signal, skip that hashed password, and move to the next hashed password.

But, returning from the signal did not cause **`crypt()`** to fail, it just went back to what it had been doing.

How we got here…

My next thought was to have **`crypt()`** run in a thread, have a separate thread handle the alarm and have it send a thread cancelation to the thread running **`crypt()`**. Signal handling with multiple threads is a bit messy, but not too bad…

Well, I never got there. I figured that if the process would not error on a signal, then the thread would also set the cancelation state to non-cancelable.

I went back to the **`crypt()`** man pages and read them very closely. I came across this statement:

> The behavior of crypt on errors isn't well standardized. **Some implementations simply can't fail** (**except by crashing the program**), others return a null pointer or a fixed string.

The claim "Some implementations simply can't fail…" is a bit of a shocker.

How we got here…

Returning to the signal handler for `SIGALRM` from the `alarm()` call, I thought I'd just have the child process **exit** with the specific exit value `EXIT_CHILD_TIMEOUT` (see `mproc_crypt.h`) when the alarm signal handler is called.

The parent process receives a `SIGCHLD` from the exited child process. The parent process then forks a new child process to pick up just past where the previous child process had left off.

After prototyping this for a while, **I defeated my foe crypt. I was able to get the behavior I wanted!**

Portland State
Computer Science

The Command Line

Required

Optional

```
# ./mproc_crypt -h
mproc_crypt p:d:o:P:N:T:n:vh
     -p passwords file name    : hashed passwords file
     -d dictionary file name   : plain text passwords file
     -o output file name       : send output to the given file name
     -P #                      : number of child processes to create
     -T #                      : number seconds to wait for a timeout
     -N #                      : increment for nice()
     -v                        : verbose output, to stderr
     -h                        : this marvelous help
```

Portland State
Computer Science

`-p passwords file name`	hashed passwords file	**REQUIRED**: The name of the file **containing hashed passwords**.
`-d dictionary file name`	plain text passwords file	**REQUIRED**: The name of the file containing **plain text passwords**.
`-o output file name`	send output to the given file name	The name of the **output file**. This will contain a list of all the cracked, failed, or timeout hash passwords.
`-P #`	number of child processes to create	The number of child processes to maintain. **If this option is not given, the default is 1 child process**.
`-T #`	number seconds to wait for a timeout	The number of seconds to wait for the **`crypt()`** function to return. **If this option is not given, the default is 5 seconds.**
`-N #`	increment for **`nice()`**	A value to add to the nice value for the process. This should be between 1 and 19.
`-v`	verbose output, to `stderr`	Supplemental output to `stderr`.
`-h`	this marvelous help	This most helpful text.

Portland State Computer Science

The `crypt()` Function

You will use the **`crypt()`** function to attempt to find a plaintext password from the dictionary file that matches the hashed password.

If you prefer to use **`crypt_r()`**, **`crypt_rn()`**, or **`crypt_ra()`**, you can. Of those, I recommend **`crypt_rn()`**.

Algorithm	Prefix		
yescrypt	`$y$`	Hash size 256 bits, Salt size up to 512 bits	Recommended for new hashes.
gost-yescrypt	`$gy$`	Hash size 256 bits, Salt size up to 512 bits	Recommended for new hashes.
bcrypt	`$2b$`	Hash size 184 bits, Salt size 128 bits	
sha512crypt	`$6$`	Hash size 512 bits, Salt size 6 to 96 bits	Acceptable for new hashes.
sha256crypt	`$5$`	Hash size 256 bits, Salt size 6 to 96 bits	Acceptable for new hashes.
md5crypt	`$1$`	Hash size 128 bits, Salt size 6 to 48 bits	It should not be used for new hashes.
descrypt	none (empty string)	Hash size 64 bits, Salt size 12 bits	It is feasible to discover any passphrase hashed with this method.
NT	`$3$`	Hash size 256 bits, Salt size 0 bits	It is feasible to discover any passphrase hashed with this method.

Portland State
Computer Science

Delimiters for the fields within the result from `crypt()`.

```
// sha256
//
// AAA:$5$rounds=5000$kUU/R2mSMwWNGRXk$/RlBqMHS6yqMQavJtYV9BDs7c2g2RBCz6LOJeuHSilD
```

The algorithm used. In this case sha256

The number of times the password was hashed by the algorithm. A higher number takes longer, making the passwords harder to crack.

The salt. In this case, it is 16 characters long, the maximum for both sha256 and sha512.

The hashed password. The plaintext password was combined with the salt and hashed 5000 times to produce this value.

Look at **`man 5 crypt`** for more information.

Portland State
Computer Science

Password	Complete hash
`chromophil`	`$y$j6T$GYnBU3lIZzRnHM3Kxc4c2pQyYqZ0Snsbd2ynLhznNVFHzWJwxg8VFiuINqDJSAhJ$LyVxq7WF93t9U7njWqmsNv.X1rNUUWPEULWTl6h9AfB`
`fifty-first`	`$gy$j6T$TBfvHz4wKmJPQ7agi8tO7ntfSMj6Vk5iqHdanfh9HEmxgWFs3U8y4vUSv62BTGTX$wB7LSA/blVrIE66dyksk3bQzSBI/CO0UR6Frj0amTnC`
`responsivity`	`$2b$09$VCtH.A/qJNpBXKyYbBs4HOzuAk.svys3TcEa2Mi7ElAFdukO39loK`
`morat`	`$6$rounds=1280$QsAPnI/2nkHVD5bU$Nr7Ez/Q9wALa1miU5uAFNzxyX1culhJeJ8ihidQ51ainjuV53ja0dQF.mt2PzuGmjDNatlF.aMjGwnIcwC0b20`
`Yurimaguas`	`$5$rounds=128$7ouhFKT5LNs0dEKQ$ff9uhIFuPDZojldu5.qh0rWU8h28LBjT41t2FyN/GO3`
`Thadeus`	`$1$FATnKM83$bfNH6X41JGSQt.QNRNXPm0`
`phreatic`	`Nu.HhPNkRyEls`
`well-exposed`	`$3$$406618de7e6c2944d063ac17caebac1a`

Look at **`man 5 crypt`** for more information.

Portland State

$1$inuEWBgy$GT/2nJOYwbmoPKAJe3sNY0
$2b$06$JOob4bRlHruiSJ2or.5souBntMTezj1AGekOQ2xz/QW1T4bYwjUtW
$5$rounds=1299$OBxQJ0BzFaviqJh$74u7k/UjWOIxQMelObtweNNMEGI7S4EbbExus9B5o6A
$5$rounds=1045$8dcnAa8r0pFTT3n$2tTBPGZLvbRhif/1w6OjY0T0BUylgMtnBCDijURyD.4
$gy$jET$ONaFAhgcoEEtvugAXs9zYMftbMjl.vDllhPRLeJn3UbQTIQpkGWL5Q.YLUsIhJQf$zX8Hx1AAinwjNvnBQXLNGGNmws4brtn1eF.BCgAqypC
$2b$18$LlsZ8BaAf6HmTaCig/vWQuk7/2hGzdYDQzkJC22AFexNJnFxhuh7G
$6$rounds=1340$lCSkiRY2mf.Avjd8$TNltcSKlwOrB19ngp8V7h9095N1hqb1fHrjyXLSRDMnufcL7Qz2exshfxL4fhZv8YaVfohg4qZQ4uhw/yz2LX/
$5$rounds=10221578$s70cQnqMoC6UJSp$kquWMsleXZRm1sDN23.lHqMRRr1c/KNLcYnGG4bXEe6
fQRHCXQCgCX32
$6$rounds=114$6XUdP1U5FHU3.HOF$80lxj6YMJDm5ODBwicXRyj6.NMV8JcCUHTcJXKW16V3zvp5yg/JJdsIzbTyPmLa92/2s.oJxfC7jcjbHl81G70
$y$j5T$dGboBU05wdzsyNPaXyhOicg68XQijcXBSDpegL80dQUl4JLYnDROGZxDL8vaogOA$1fUoYNDkd5JE8qqXeRRlUV15LaWtFT27TnacigV8vq6
$1$DjzPB0rn$SMmYjwa5MBG/.PY/eCZqu.

Portland State
Computer Science

The Result for Each Hashed Password

For each hashed password, you will need to test the hashed password against words in the dictionary.

For each hashed password, there are three possible outcomes:

1. **The plaintext password in found in the dictionary**. Your program will output showing matched password with a message:
```
cracked <hashed password> <the plaintext password>
cracked Zhs3bK/eCcaqU makomako
```
2. **The plaintext password is not in the dictionary**. Your program will output showing the failed hashed password with a message:
```
failed <hashed password>
failed XqLt8P7XfCkbI
```
3. When attempting to crack a hashed password, **a timeout occurs, forcing the child process to exit**. Your program will output showing the timeout with a message:
```
timeout <hashed password>
timeout $2b$19$Y/0YEfI9CWNc8WXqA4wDguRAUABGQV5rSk9p12WWjtg1cYPVjQkSi
```

These files can be found in

```
~rchaney/Classes/cs333/Labs/Lab3
```

on babbage only. Both the passwords file and dictionary file are required to be provided on the command line.

```
./mproc_crypt -p mpasswords10.txt -d mdictionary10.txt -P 2
> LOG.out  2> LOG.err
```

Parent and child process statistics are sent to `stderr`.

Specify the number of child processes to use to process the passwords.

- **Don't use more than 10 child processes at a time.**
- **It will very likely require more than 2 total child processes to complete.**

Capture `stdout` into the file `LOG.out`. This is an alternarive to using the `-o output_filename`.

```
# cat LOG.out
cracked $2b$06$JOob4bRlHruiSJ2or.5souBntMTezj1AGekOQ2xz/QW1T4bYwjUtW fetus
timeout $2b$18$LlsZ8BaAf6HmTaCig/vWQuk7/2hGzdYDQzkJC22AFexNJnFxhuh7G
cracked
$6$rounds=1340$lCSkiRY2mf.Avjd8$TNltcSKlwOrB19ngp8V7h9095N1hqb1fHrjyXLSRDMnufcL7Qz
2exshfxL4fhZv8YaVfohg4qZQ4uhw/yz2LX/ randomwise
cracked fQRHCXQCgCX32 indictee
timeout
$5$rounds=10221578$s70cQnqMoC6UJSp$kquWMsleXZRm1sDN23.lHqMRRr1c/KNLcYnGG4bXEe6
cracked
$6$rounds=1144$6XUdP1U5FHU3.HOF$80lxj6YMJDm5ODBwicXRyj6.NMV8JcCUHTcJXKW16V3zvp5yg/
JJdsIzbTyPmLa92/2s.oJxfC7jcjbHl81G70 vasomotory
cracked
$y$j5T$dGboBU05wdzsyNPaXyhOicg68XQijcXBSDpegL80dQUl4JLYnDROGZxDL8vaogOA$1fUoYNDkd5
JE8qqXeRRlUV15LaWtFT27TnacigV8vq6 Bullivant
failed $1$DjzPB0rn$SMmYjwa5MBG/.PY/eCZqu.
cracked $1$inuEWBgy$GT/2nJOYwbmoPKAJe3sNY0 Gesnera
cracked $5$rounds=1299$OBxQJ0BzFaviqJh$74u7k/UjWOIxQMelObtweNNMEGI7S4EbbExus9B5o6A
Bullivant
failed
$gy$jET$ONaFAhgcoEEtvugAXs9zYMftbMjl.vDllhPRLeJn3UbQTIQpkGWL5Q.YLUsIhJQf$zX8Hx1AAi
nwjNvnBQXLNGGNmws4brtn1eF.BCgAqypC
```

Portland State

```
# cat LOG.out
cracked $2b$06$JOob4bRlHruiSJ2or.5souBntMTezj1AGekOQ2xz/QW1T4bYwjUtW fetus
timeout $2b$18$LlsZ8BaAf6HmTaCig/vWQuk7/2hGzdYDQzkJC22AFexNJnFxhuh7G
cracked
$6$rounds=1340$lCSkiRY2mf.Avjd8$TNltcSKlwOrB19ngp8V7h9095N1hqb1fHrjyXLSRDMnufcL7Qz
2exshfxL4fhZv8YaVfohg4qZQ4uhw/yz2LX/ randomwise
cracked fQRHCXQCgCX32 indictee
timeout
$5$rounds=10221578$s70cQnqMoC6UJSp$kquWMsleXZRm1sDN23.lHqMRRr1c/KNLcYnGG4bXEe6
cracked
$6$rounds=1144$6XUdP1U5FHU3.HOF$80lxj6YMJDm5ODBwicXRyj6.NMV8JcCUHTcJXKW16V3zvp5yg/
JJdsIzbTyPmLa92/2s.oJxfC7jcjbHl81G70 vasomotory
cracked
$y$j5T$dGboBU05wdzsyNPaXyhOicg68XQijcXBSDpegL80dQUl4JLYnDROGZxDL8vaogOA$1fUoYNDkd5
JE8qqXeRRlUV15LaWtFT27TnacigV8vq6 Bullivant
failed $1$DjzPB0rn$SMmYjwa5MBG/.PY/eCZqu.
cracked $1$inuEWBgy$GT/2nJOYwbmoPKAJe3sNY0 Gesnera
cracked $5$rounds=1299$OBxQJ0BzFaviqJh$74u7k/UjWOIxQMelObtweNNMEGI7S4EbbExus9B5o6A
Bullivant
failed
$gy$jET$ONaFAhgcoEEtvugAXs9zYMftbMjl.vDllhPRLeJn3UbQTIQpkGWL5Q.YLUsIhJQf$zX8Hx1AAi
nwjNvnBQXLNGGNmws4brtn1eF.BCgAqypC
```

Portland State

```
# cat LOG.out
cracked $2b$06$JOob4bRlHruiSJ2or.5souBntMTezj1AGekOQ2xz/QW1T4bYwjUtW fetus
timeout $2b$18$LlsZ8BaAf6HmTaCig/vWQuk7/2hGzdYDQzkJC22AFexNJnFxhuh7G
cracked
$6$rounds=1340$lCSkiRY2mf.Avjd8$TNltcSKlwOrB19ngp8V7h9095N1hqb1fHrjyXLSRDMnufcL7Qz
2exshfxL4fhZv8YaVfohg4qZQ4uhw/yz2LX/ randomwise
cracked fQRHCXQCgCX32 indictee
timeout
$5$rounds=10221578$s70cQnqMoC6UJSp$kquWMsleXZRm1sDN23.lHqMRRr1c/KNLcYnGG4bXEe6
cracked
$6$rounds=1144$6XUdP1U5FHU3.HOF$80lxj6YMJDm5ODBwicXRyj6.NMV8JcCUHTcJXKW16V3zvp5yg/
JJdsIzbTyPmLa92/2s.oJxfC7jcjbHl81G70 vasomotory
cracked
$y$j5T$dGboBU05wdzsyNPaXyhOicg68XQijcXBSDpegL80dQUl4JLYnDROGZxDL8vaogOA$1fUoYNDkd5
JE8qqXeRRlUV15LaWtFT27TnacigV8vq6 Bullivant
failed $1$DjzPB0rn$SMmYjwa5MBG/.PY/eCZqu.
cracked $1$inuEWBgy$GT/2nJOYwbmoPKAJe3sNY0 Gesnera
cracked $5$rounds=1299$OBxQJ0BzFaviqJh$74u7k/UjWOIxQMelObtweNNMEGI7S4EbbExus9B5o6A
Bullivant
failed
$gy$jET$ONaFAhgcoEEtvugAXs9zYMftbMjl.vDllhPRLeJn3UbQTIQpkGWL5Q.YLUsIhJQf$zX8Hx1AAi
nwjNvnBQXLNGGNmws4brtn1eF.BCgAqypC
```

```
# cat LOG.out
cracked $2b$06$JOob4bRlHruiSJ2or.5souBntMTezj1AGekOQ2xz/QW1T4bYwjUtW fetus
timeout $2b$18$LlsZ8BaAf6HmTaCig/vWQuk7/2hGzdYDQzkJC22AFexNJnFxhuh7G
cracked
$6$rounds=1340$lCSkiRY2mf.Avjd8$TNltcSKlwOrB19ngp8V7h9095N1hqb1fHrjyXLSRDMnufcL7Qz
2exshfxL4fhZv8YaVfohg4qZQ4uhw/yz2LX/ randomwise
cracked fQRHCXQCgCX32 indictee
timeout
$5$rounds=10221578$s70cQnqMoC6UJSp$kquWMsleXZRm1sDN23.lHqMRRr1c/KNLcYnGG4bXEe6
cracked
$6$rounds=1144$6XUdP1U5FHU3.HOF$80lxj6YMJDm5ODBwicXRyj6.NMV8JcCUHTcJXKW16V3zvp5yg/
JJdsIzbTyPmLa92/2s.oJxfC7jcjbHl81G70 vasomotory
cracked
$y$j5T$dGboBU05wdzsyNPaXyhOicg68XQijcXBSDpegL80dQUl4JLYnDROGZxDL8vaogOA$1fUoYNDkd5
JE8qqXeRRlUV15LaWtFT27TnacigV8vq6 Bullivant
failed $1$DjzPB0rn$SMmYjwa5MBG/.PY/eCZqu.
cracked $1$inuEWBgy$GT/2nJOYwbmoPKAJe3sNY0 Gesnera
cracked $5$rounds=1299$OBxQJ0BzFaviqJh$74u7k/UjWOIxQMelObtweNNMEGI7S4EbbExus9B5o6A
Bullivant
failed
$gy$jET$ONaFAhgcoEEtvugAXs9zYMftbMjl.vDllhPRLeJn3UbQTIQpkGWL5Q.YLUsIhJQf$zX8Hx1AAi
nwjNvnBQXLNGGNmws4brtn1eF.BCgAqypC
```

In addition to generating output for each processed hashed password to **`stdout`** (or the output file), you need to output summary information to **`stderr`**. The summary information is for each child process and the sum for all of the processes. Shown below is an example:

```
# cat LOG.err
  child 1390079 cracked: 1 failed: 0 timeout: 1 total: 2
  child 1390138 cracked: 2 failed: 0 timeout: 1 total: 3
  child 1390147 cracked: 2 failed: 1 timeout: 0 total: 3
  child 1390078 cracked: 3 failed: 1 timeout: 0 total: 4
PARENT cracked: 8 failed: 2 timeout: 2 total: 12
```

- The output for the child processes is the child pid, # cracked, # failed, and # timeouts. A child process will never have more than 1 timeout.
- The parent process shows the sum of the values from all the child processes.

Use indentation to make it easier to read the output.

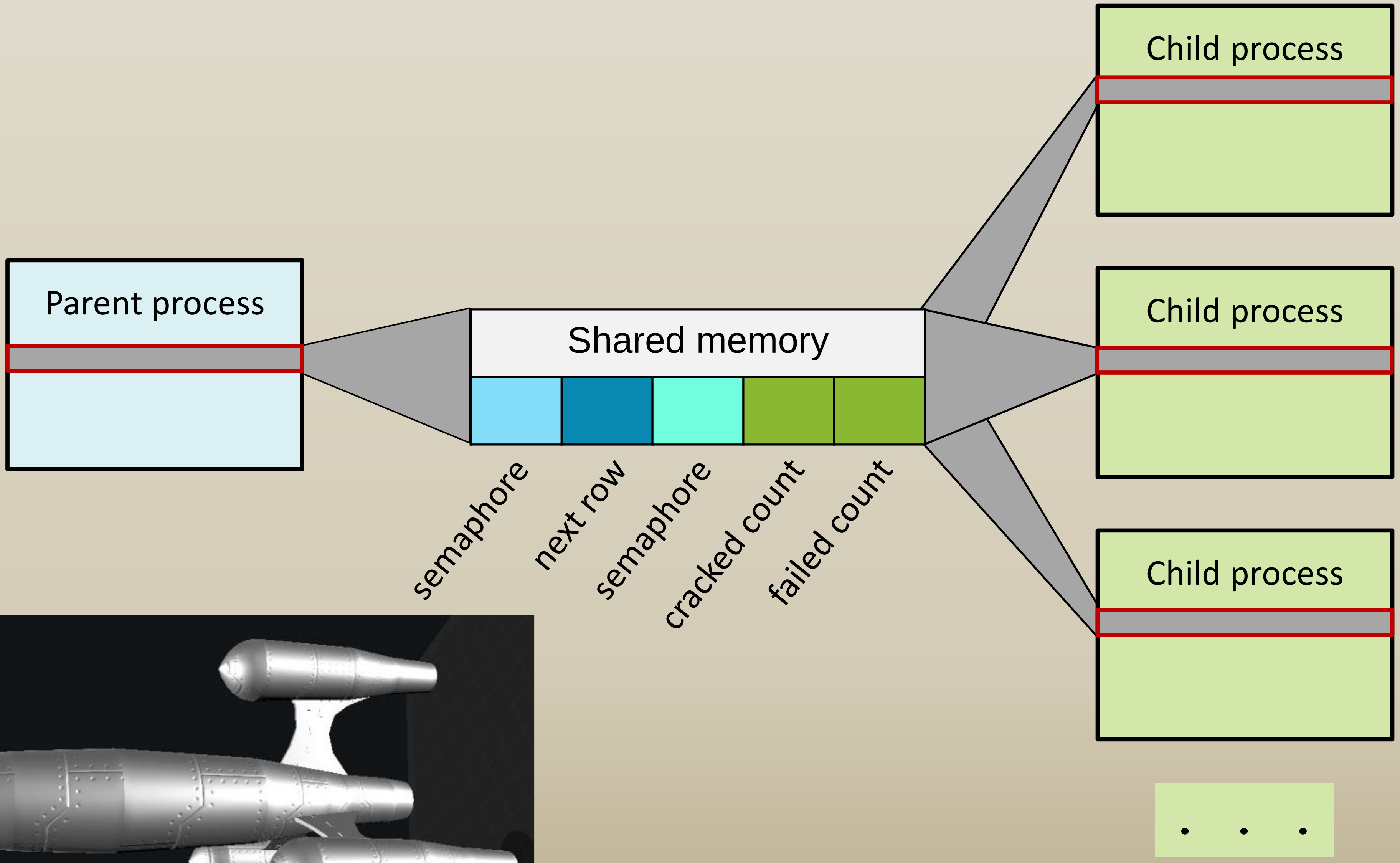
The Shared Memory Segment

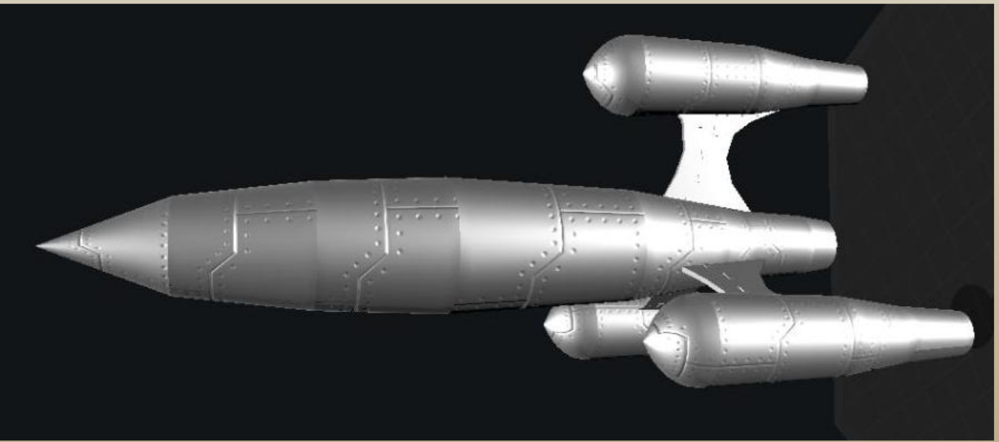
How do you keep track of which hashed passwords have been processed and how many hashed passwords have been cracked or failed?

It’s simple, **establish a shared memory segment that the parent process and all child processes can access**. Use semaphores to protect the shared values.

The parent process creates the shared memory segment and the child processes inherit it.

The Shared Memory Segment





```c
while(fgets(buf, BUF_SIZE, stdin) != NULL) {
      // The format of input is:
       //    plaintext:hashed
       plaintext = strtok(buf, ":");
       if (argc > 1) {
            plaintext[0] = 'q';
       }
       setting = strtok(NULL, "\n");
       crypt_return = crypt(plaintext, setting);
       printf("%s\t%s\t%s\t", plaintext, setting, crypt_return);
       if (strcmp(crypt_return, setting) != 0) {
            printf("bad\n");
       }
       else {
            printf("good\n");
       }
}
```

The **`uncrypt_example.c`** file in the Lab directory.

Portland State
Computer Science

```
# ./uncrypt_example < key10.txt
missuppose
$6$rounds=1232$RFATGMcc5YXCtZ/m$WnnflEKQ6UmeHWodtvxCaLlaW2kyQ4cbo9w98HQDsApY.Ts
FznKNGaarT2oj/Xb3NhKINlA9okwrmaQGD2hn4/
$6$rounds=1232$RFATGMcc5YXCtZ/m$WnnflEKQ6UmeHWodtvxCaLlaW2kyQ4cbo9w98HQDsApY.Ts
FznKNGaarT2oj/Xb3NhKINlA9okwrmaQGD2hn4/    good
nonjuress
$y$j5T$quJ9xHl48hctoihF0pCh3gUj/P2JFyFnsVLRcxcamAiIf3mX66if5k/RtaFpE7O2$rWOzCtw
fCk4K4by8AbIlvDiINgvdX4zxE7CP9lcHjV2
$y$j5T$quJ9xHl48hctoihF0pCh3gUj/P2JFyFnsVLRcxcamAiIf3mX66if5k/RtaFpE7O2$rWOzCtw
fCk4K4by8AbIlvDiINgvdX4zxE7CP9lcHjV2      good
eye-brightening $3$$67816a38595fc99f6964276d48161a67
$3$$67816a38595fc99f6964276d48161a67    good
indictee        $1$.ygLbqdg$cl0uJNT/KnD8mTr/2Sm9T0
$1$.ygLbqdg$cl0uJNT/KnD8mTr/2Sm9T0      good
biochron        YZ9yYUyKJaSt6   YZ9yYUyKJaSt6   good
```

Portland State
Computer Science

Rough Outline of the Parent Code

Parent process

- Creates the shared memory segment, initializes the values
- Allocates any other memory that all processes will use (hashes and dictionary)
- Establishes a signal handler for **`SIGCHLD`**. When a child process exits, this signal is delivered to the parent process.
- I established signal handlers for the following signals: **`SIGINT`**, **`SIGQUIT`**, **`SIGTERM`**, and **`SIGHUP`**. These are so the parent process can perform cleanup if it receives one of those signals.
- Establishes an exit handler to deallocate memory and remove shared memory segment. I used **`atexit()`**.
- Forks the correct number child processes
- Enters loop calling **`pause()`**

Portland State
Computer Science

Rough Outline of the Parent Code

Parent process exit handler

- Send a SIGTERM to each process. I keep an array of all the child pids and send a `kill()` to each. Reap each child process.
- Unmap and unlink the shared memory segment.
- Deallocate memory for the hashed passwords and dictionary (ragged arrays)
- Free the child pids array.
- Close the output file.

Portland State
Computer Science

Rough Outline of the Parent Code

Parent process SIGCHLD handler

- Reap the child process. I used `waitpid()` for this.
- Get the exit value, using the `WEXITSTATUS()` macro to trim out extra bits.
- If the exit value is `EXIT_CHILD_TIMEOUT`, increment the counter for timeouts and fork a new child process.
- If the exit value is `EXIT_SUCCESS`, decrement the counter for number of the pool of active child processes.
- When the number of processes in the pool of child processes reaches zero, exit the parent process. I call `exit(EXIT_SUCCESS)`.

Portland State Computer Science

Rough Outline of the Child Code

Child process

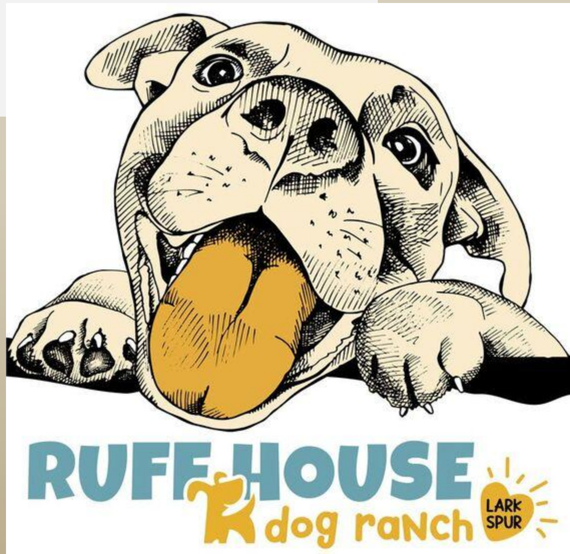
- I do not call **`exec()`** in the child processes.
- Establish a signal handler for **`SIGALRM`**.
- Begin loop of getting next password to process (using the shared memory to get the next element in the array of hashed passwords).
  - Begin loop of looking through the dictionary
    - Before calling **`crypt()`**, I call **`alarm(timeout)`**.
    - Call **`crypt()`** using the hashed password and the current word from the dictionary.
    - Call **`alarm(0)`** to disable all pending alarms. If the alarm expired before calling **`alarm(0)`**, the **`SIGALRM`** signal handler is called.
    - If **`crypt()`** returns a match, the password is cracked. Output a message **cracked** to output file and break out of the dictionary loop.
  - If the password was not cracked, output a **failed** message to output file.

Rough Outline of the Child Code

Child process (continued)

- Get the next hashed password and begin looking through the dictionary again.

- When all hashed passwords have been processed, output a summary of cracked and failed passwords for this child process to **`stderr`**.
- If the child process reaches this point, it will not have had a timeout. The **`SIGALRM`** handler will have been called.
- Update the global statistics for the password cracking in the shared memory segment.
- Exit with **`EXIT_SUCCESS`**.



Portland State
Computer Science

Rough Outline of the Child Code

Child process SIGALRM handler

- Update the global statistics for the password cracking in the shared memory segment.
- Output a message **timeout** to the output file.
- Output a summary of cracked and failed passwords for this child process to **`stderr`**. This child will have a timeout value of 1.
- Call **`exit()`** with a value of **`EXIT_CHILD_TIMEOUT`**. It is important that this value be used to pass information to the parent process that the child exited from a timeout and a new child process be started. The parent process will use this value in its **`SIGCHLD`** handler.

Portland State
Computer Science

Data Shared between Parent and Child

I create a ragged array of the hashed password and a ragged array of the dictionary. This allows me to easily step through the arrays.

Since the address space of the parent process is inherited by each child process, each child process has the same ragged arrays.

When the parent process creates the shared memory segment, the child processes inherit the shared memory segment.

Portland State
Computer Science

Memory Leaks

While I encourage you to free any memory you allocated in the parent process (which may be inherited by child processes) and any additional memory allocated in child processes, **you will have memory leaks**.

The abrupt and graceless termination of a process with a timeout leaves **`crypt()`** unable to cleanup the memory it allocated.

And There Was Much Rejoicing