Tutorial 4

Process and Threads

fork system call

```
#include <unistd.h>

pid_t fork(void);
```

fork() creates a new process by duplicating the calling process.

The new process is referred to as the *child* process.

On success, the PID of the child process is returned in the parent, and 0 is returned in the child. On failure, -1 is returned in the parent, no child process is created.

man fork [Also at https://man7.org/linux/man-pages/man2/fork.2.html]

exec system call

```
#include <unistd.h>
```

The **exec**() family of functions replaces the current process image with a new process image.

execl, execlp, execle, execv, execvp, execvpe

The **exec**() functions return only if an error has occurred. The return value is -1.

man exec [Also at https://man7.org/linux/man-pages/man3/exec.3.html]

clone system call

```
#define _GNU_SOURCE

#include <sched.h>

int clone(int (*fn)(void *), void *stack, int flags, void
*arg, ...);
```

Creates a new (“child”) process, in a matter similar to fork. This provides more precise control over what pieces of execution context are shared between the calling process and the child process. For example, the caller can control whether or not the two processes share the virtual address space, the table of file descriptors, and the table of signal handlers.

Like others, returns -1 on failure.

man clone

Getting confused?

This gives a good overview of these all, and why you would prefer one over the above:
https://stackoverflow.com/questions/4856255/the-difference-between-fork-vfork-exec-and-clone

This is an excellent dive into Linux semantics over process and thread creation:
https://eli.thegreenplace.net/2018/launching-linux-threads-and-processes-with-clone

Threads

`pthreads - POSIX threads`

POSIX.1 specifies a set of interfaces (functions, header files) for threaded programming commonly known as POSIX threads, or Pthreads.

Difference between process and threads?

pthread_create(): create a new thread

```
int pthread_create(pthread_t *restrict thread,
                   const pthread_attr_t *restrict attr,
                   void *(*start_routine)(void *),
                   void *restrict arg);
```

man pthreads

https://man7.org/linux/man-pages/man7/pthreads.7.html

Demo.

Mini-Problem [not to worry, ungraded :)]

cat $file_1$ $file_2$ .... $file_n$

This prints the contents of all the files together on the standard output. However, it becomes tough to find where a file ends and the next begins.

So, your task will be to make a utility program better_cat which will take a list of files as arguments and for each file, print the name and then the contents.

While this is fully doable by yourself, for printing a file, you have to use cat.

Thus, you will be printing the name and then creating a new cat process (to print the contents) for every file in arguments.