```
  - ‘git am /path/to/a1-zones-base.patch‘ in /usr/src
- Build the kernel
  - ‘cd /usr/src/sys/arch/amd64/compile/GENERIC.MP‘
  - ‘make obj‘
  - ‘make config‘
  - ‘make -j 5‘
  - ‘doas make install‘
- Reboot into the kernel
  - ‘doas reboot‘
- ‘make obj‘ in /usr/src
- ‘doas make includes‘ in /usr/src/include
  - Verify the zones syscalls are in /usr/include/sys/syscall.h
  - Verify /usr/include/sys/zones.h exists
- Make and install libc
  - ‘cd /usr/src/lib/libc‘
  - ‘make -j 5‘
  - ‘doas make install‘
- Optional: make ps, and pkill/pgrep
- make zone(8)
  - ‘cd /usr/src/us
  - ‘make‘
  - ‘doas make inst
- Verify ‘zone(8)‘
$ zone list
      ID NAME
       0 global
$ zone create
usage: zone create
$ zone create test
zone: create: Opera
$ doas zone create
doas (dlg@comp3301.
$ zone list
      ID NAME
       0 global
   42101 test
$ zone id
0
$ zone id test
42101
$ zone exec test ps
zone: enter: Operation not permitted
$ doas zone exec test ps -aux
USER       PID %CPU %MEM    VSZ    RSS TT  STAT    STARTED        TIME COMMAND
root     41705  0.0  0.1    628    580 p0  R+pU/0  3:37PM     0:00.14 ps -aux
$ doas zone exec test zone id
42101
$ doas zone exec test zone id global
zone: id: No such process
$
```

As you add the functionality specified in the next sections, some of these steps will be repeated. eg, changing the kernel means rebuilding and installing the kernel. Adding a syscall means making the syscall stub as a function visible in the headers (make includes), and callable through libc.

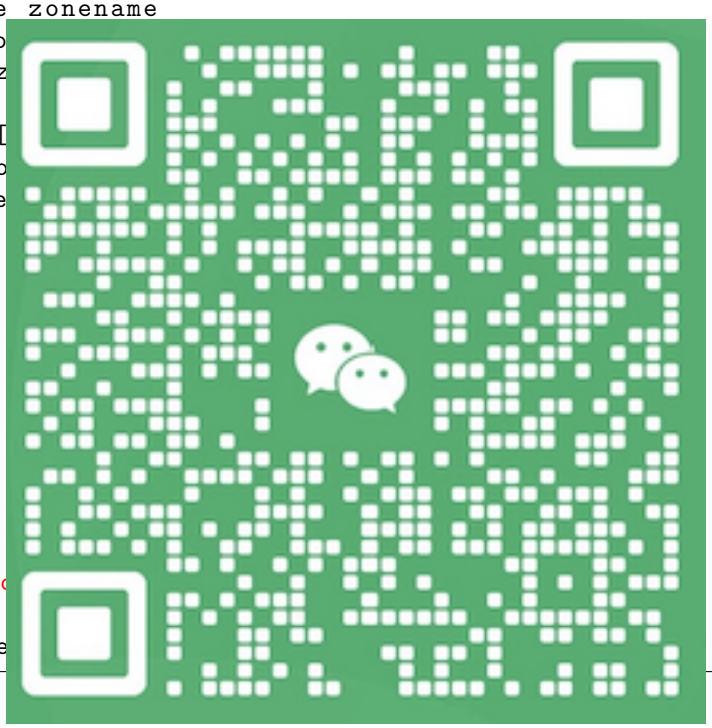
A note on errors

We have over-specified the errors you should return from your syscalls - if you do not require an error code (for example, never returning `ENOMEM` on memory failures because you never allocate any memory) then you do not have to use it. The reverse is also true - if you find an error case that is not listed, choose an appropriate error from `errno(2)`. We will not explicitly test all errors, but during your code interview, we will expect you to be able to explain the suitability of the error codes you use.

4.2 Zone Rename

The `zone(8)` commands should be extended to enable renaming of zones. Zones should only be able to be renamed by the owner, root, or members of the zone's group. Additionally, the global zone cannot be renamed, and zone names must be unique.

```
$ zone
usage:  zone create zonename
        zone destro
        zone exec z
        zone list
        zone name [
        zone id [zo
        zone rename
$ doas zone create
$ zone list
      ID NAME
       0 global
     289 foo
$ doas zone rename
$ zone list
      ID NAME
       0 global
     289 bar
$ doas zone rename
zone: rename:  Invali
$ doas zone rename
zone: rename: File e
```



4.3 Modifications to Existing Syscalls

`zone_create()` syscall

The `zone_create()` syscall should now ensure that the created zone is associated with the group of the user that created it, as well as the user themself. Additionally, this will mean ensuring that non-root users can create zones. The definition of `zone_create()` should not change - it should still take a single `char *zonename` as its argument.

All other syscalls

The full suite of `zone_*` syscalls should permit users with matching credentials (owner or group) to perform zone operations on them, not only the root user. The credentials may be changed so appropriate synchronisation should be used. Namely, we expect that, unless credentials are being changed by another thread, authorisation should be non-blocking.

4.4 Zone name and zone list

`zone_name()` syscall

The `zone_name()` syscall should be renamed to `zone_info()`. Subsequently, it should return not only the name and namelen, but also the zone, user and group id, preferably all bundled in a struct format. However you may pass back one or more of these as individual parameters if that is easier. The `zone(8)` userland sub-command for `zone name` should also be modified in line with these changes - the name should be changed to `zone info` and the additional information should be provided to the user. Alternatively, you may also create zone info as an independent command.

`zone list`

The `zone list` subcommand should now take flags: `-o` and `-g`. If the `-o` flag is provided, the owner of the zone should be printed, and if the `-g` flag is provided, the zone's group should be printed. If both flags are provided, print both. The extra fields should be printed as extra columns in the current table format. zone id and name must be displayed first. However, the order of the additional fields does not matter.

4.5 Zone chown

The `zone(8)` command to enable changing the owner and group of le to be changed by the owner, root, or me r of the global zone cannot be changed.

```
$ zone
usage:  zone create
        zone destro
        zone exec z
        zone list
        zone name [
        zone id [zo
        zone chown
        zone chgrp
```

The two subcommands . `zone chown` takes the name of a zone an er to the user with the specified name. If a zone with the name `zonename` does not exist, `zone(8)` will attempt to interpret the argument as a numeric zone identifier.

`zone chgrp` behaves similarly, but instead, it uses the `zone_chgrp()` syscall to change the zone's group to the specified group name.

To support these subcommands, you will need to implement the following system calls:

`zone_chown()` syscall

```
int     zone_chown(zoneid_t z, uid_t user);
```

The `zone_chown()` syscall alters the owner of the zone identified by the `z` argument. The new owner should be the owner identified by the `user` argument. If called from a non-global zone, then the `z` id must be the identifier for the current zone, but in the global zone, it can be any zone identifier. This means that to the user, a non-global zone should only be able to see itself.

Potential Errors:

- EPERM - the user does not have permission to alter the zone `z`
- ESRCH - the zone identified by `z` does not exist
- ENOMEM - the system was not able to allocate memory
- EINVAL - the zone to alter was the global zone

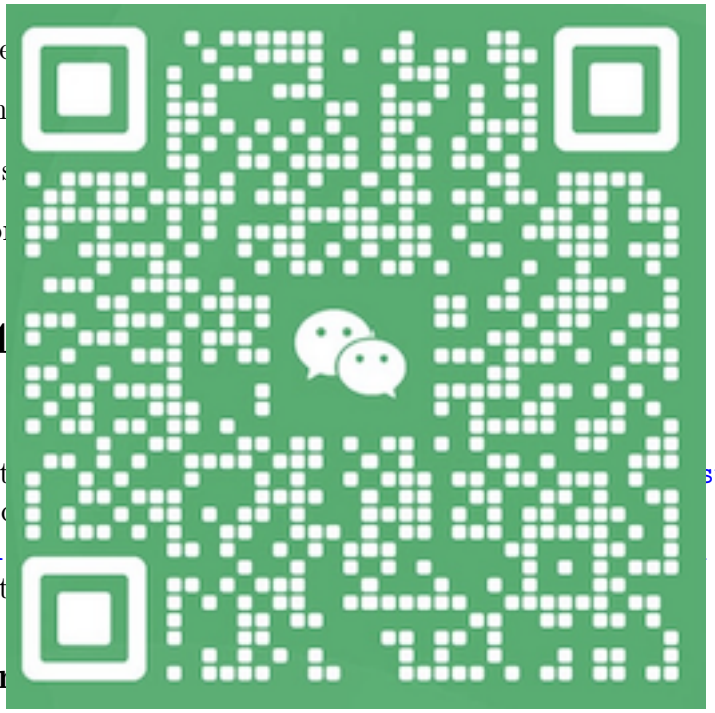
`zone_chgrp()` syscall

```
int     zone_chgrp(zoneid_t z, gid_t group);
```

The `zone_chgrp()` syscall alters the owner of the zone identified by the `z` argument. The new owner should be the group identified by the `group` argument. If called from a non-global zone, then the `z` id must be the identifier for the current zone, but in the global zone, it can be any zone identifier. This means that to the user, a non-global zone should only be able to see itself.

Potential Errors:

- EPERM - the use
- ESRCH - the zon
- ENOMEM - the s
- EINVAL - the zo



5 Other Req

5.1 Code Style

Your code is to be writt `style(9)` man page.
An automatic tool for
https://stluc.manta
This tool will be used t

5.2 Compilatio

Your code for this assignment is to be built on an `amd64` OpenBSD 7.5 system identical to your course-provided VM.

The following steps must succeed:

- `make obj; make config; make` in src/sys/arch/amd64/compile/GENERIC.MP
- `make obj; make includes` in src
- `make obj; make; make install` in src/lib/libc
- `make obj; make; make install` in src/usr.sbin/zone

The existing `Makefile`s in the provided code are functional as-is, but may need modification as part of your work for this assignment. Note that the existing Makefile ensures the `-Wall` flag is passed to the compiler, as well as a few other warning and error-related flags.

5.3 Provided code

The provided code, which forms the basis for this assignment, can be downloaded as a single patch file at:
https://stluc.manta.uqcloud.net/comp3301/public/2024/a1-zones-base.patch

You should create a new `a1` branch in your repository based on the `openbsd-7.5` tag using `git checkout`, and then apply this base patch using the `git am` command:

```
$ git checkout -b a1 openbsd-7.5
$ ftp https://stluc.manta.uqcloud.net/comp3301/public/2024/a1-zones-base.patch
$ git am < a1-zones-base.patch
$ git push origin a1
```

5.4 Recommendations

The following order wil s assignment:

1. Download, build,
2. Add the `zone re`
3. Minimally modify
4. Rewrite `zone_nar`
   This ensures you
5. Add the `zone_ch`
6. Add the correspo `one(8)`.
7. Fix up any tiny b going... right?

Additionally, it is stron t the following APIs be considered for use a:

- `sys/ucred.h` - pr group credentials
- `copyin(9)`/`copyout(9)` - provides the ability to copy data across the userspace boundary
- `user_from_uid(3)` - conversions from group/user name to id and back
- `strtonum(3)` - BSD style safe string to int conversions
- Finally, you may wish to look at the header file `sys/proc.h` to see how user and group credentials are currently stored by threads.