kworkflow

Implementing lore.kernel.org interface

GSoC 2023 Contributor Application

David de Barros Tadokoro
University of São Paulo
email: davidbtadokoro@gmail.com
GitHub: https://github.com/davidbtadokoro

About me

My name is David Tadokoro, and I'm a Computer Science student from Brazil. I'm really interested in all things related to computers. Still, in the last year or so, I became fascinated with two topics: Operating Systems and Open Source communities.

Lately, my main programming languages are Bash, Java, and Python, but I'm trying to enhance my C knowledge.

I am a member of the FLOSS Competence Center at the University of São Paulo (CCSL-USP). There, I can access some machines to make any development or test for the kworkflow project.

Why I chose kworkflow?

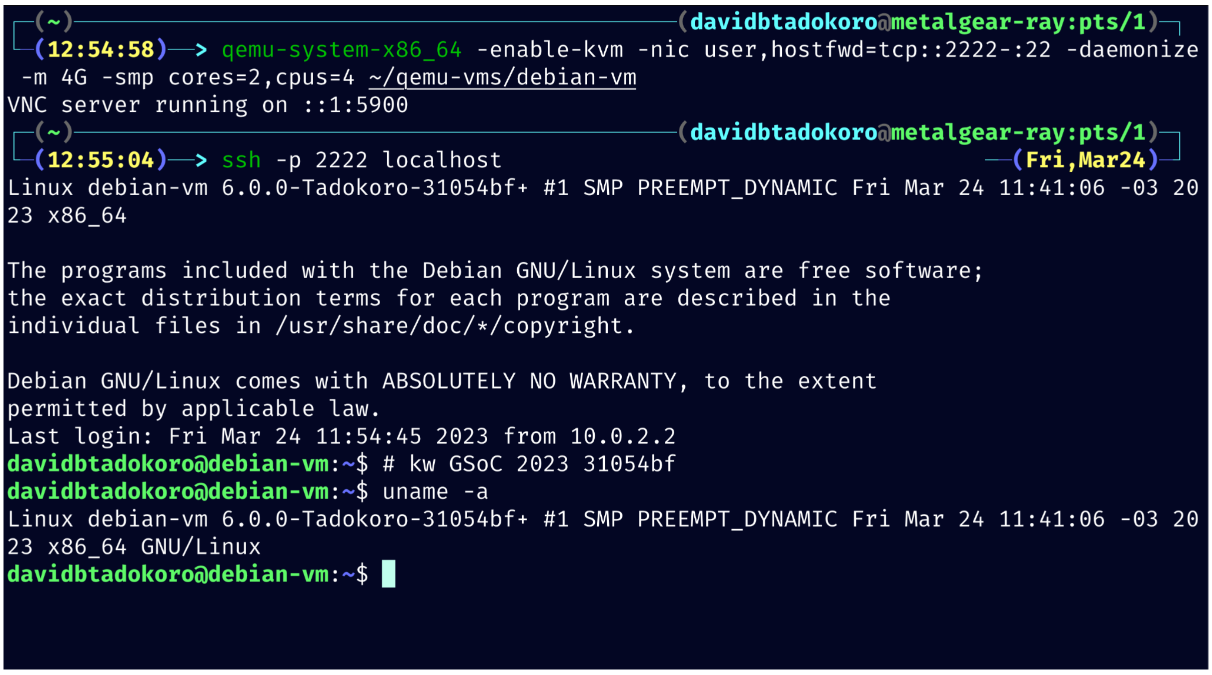
Nowadays, the Linux kernel is a ubiquitous and critical piece of software for the modern world, as it has been for years. Kernel development has a giant impact on the technology industry as a whole. The kw project aims to provide tools for everyday tasks and be a unified environment for kernel developers.

Since I became involved in the kw project, I learned a lot about working with a community (which I find the most invaluable), the Linux kernel, bash, git, and the kw project.

Prerequisites

As a well-thought exercise to warm-up applicants, we needed to complete tasks related to the problems the kw project aims to solve, how it solves some of them, and ways developers can contribute to solving more of them.

I first had to compile and install a simple modified version of the Linux kernel using virtual machines (in this case, a Debian system). This step is one of the core problems that kw solves and is always fun. The following steps lay the foundation for contributing to the project, like reading the bash manual, learning good practices, and understanding the coding style. All of those are fundamental and should always be referred to while contributing.



```
┌─(~)──────────────────────────────────────────────────(davidbtadokoro@metalgear-ray:pts/1)─┐
└─(12:54:58)──> qemu-system-x86_64 -enable-kvm -nic user,hostfwd=tcp::2222-:22 -daemonize
 -m 4G -smp cores=2,cpus=4 ~/qemu-vms/debian-vm
VNC server running on ::1:5900
┌─(~)──────────────────────────────────────────────────(davidbtadokoro@metalgear-ray:pts/1)─┐
└─(12:55:04)──> ssh -p 2222 localhost                                        ──(Fri,Mar24)─┘
Linux debian-vm 6.0.0-Tadokoro-31054bf+ #1 SMP PREEMPT_DYNAMIC Fri Mar 24 11:41:06 -03 20
23 x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Fri Mar 24 11:54:45 2023 from 10.0.2.2
davidbtadokoro@debian-vm:~$ # kw GSoC 2023 31054bf
davidbtadokoro@debian-vm:~$ uname -a
Linux debian-vm 6.0.0-Tadokoro-31054bf+ #1 SMP PREEMPT_DYNAMIC Fri Mar 24 11:41:06 -03 20
23 x86_64 GNU/Linux
davidbtadokoro@debian-vm:~$ █
```

Figure 1: Proof of the warm-up exercise

Above is the screenshot of the Debian QEMU virtual machine with a Linux kernel version 6.0.0 modified, compiled, and installed for the exercise. The modification was to change the kernel's name to 6.0.0-Tadokoro-31054bf (the number suffix was the ID given to me by my project's mentor).

Contributions to kworkflow

Contribution	Status
tests: report_test: Fix terminal and file outputs from test_save_data_to()	Accepted
documentation: man: kw: Revise deploy subsection	Accepted
src: kw_remote: Fix not failing when missing valid options	Accepted
src: kw_remote: Fix remove remote that is prefix of other remote	Accepted
documentation: man: remote: Add --set-default option to kw remote man	Accepted
documentation: man: remote: Revise kw remote man page	Accepted
src: deploy: Fix individual redirects to same file	Accepted
src: deploy: Fix unreachable command	Accepted
tests: deploy_test: Fix incorrect ShellCheck fail	Accepted

Contribution	Status
src: deploy: Implement deploy commands outside kernel tree	Accepted
src: _kw: Add ZSH completion for kw (*series of 28 commits*)	Accepted
documentation: content: project_structure: Add Bash and Zsh completions instructions	Accepted
documentation: dependencies: Add curl and xpath dependencies	Accepted
src: upstream_patches_ui: Add help option	Accepted
src: upstream_patches_ui: Fix list_patches menu title	Accepted
src: upstream_patches_ui: Fix Dashboard screen message box	Accepted
src: upstream_patches_ui: Add loading screen for delayed actions	Accepted
tests: kwlib_test: Suppress wrong ShellCheck fail	Accepted
src: kwib: Add function for safety checking paths	Accepted
src: upstream_patches_ui: Add bookmark feature	Accepted
src: upstream_patches_ui: Add message box screen	Accepted
src: lib: dialog_ui: Add persistence to checklist options	Accepted
src: upstream_patches_ui: Add apply patch action	Work in Progress

Contributions to the Linux kernel (AMD repository)

To better understand the workflow of Linux kernel developers, I submitted three patches to the AMD repository.

Contributions	Status
drm/amd/display: add prefix to amdgpu_dm_plane.h functions	Accepted
drm/amd/display: remove legacy fields of dc_plane_cap struct	Accepted
drm/amd/display: add prefix to amdgpu_dm_crtc.h functions	Accepted

Link to the kernel source tree: https://gitlab.freedesktop.org/agd5f/linux

Project Proposal

The Linux kernel is collaboratively developed using mailing lists. Currently, kw already has a rich feature to aid in formatting and sending patches through email, but only a prototype for dealing with the recipient side of the patches. Actions like consulting the mailing lists, reviewing/replying patches, applying them, and building and deploying the kernel patch version are recurrent tasks for maintainers/contributors and anyone involved in the Linux kernel community.

With this in mind, my GSoC proposal aims to refine and expand `kw upstream-patches-ui`, which is the feature responsible for providing an interface with the lore archives of the mailing lists related to the Linux kernel (lore.kernel.org) and tools for dealing with the tasks listed above. In particular, I aim to allow users to use the feature in their patch review routine, letting them do inline reviews, reply with Reviewed-by, and much more. Also, the current user interface used for the feature is dialog, and I would like to at least experiment with using other interfaces like a web interface.

`kw upstream-patches-ui` starting point:

Below are screenshots of the starting point of the feature (before my contributions) and some interesting changes that can be done.

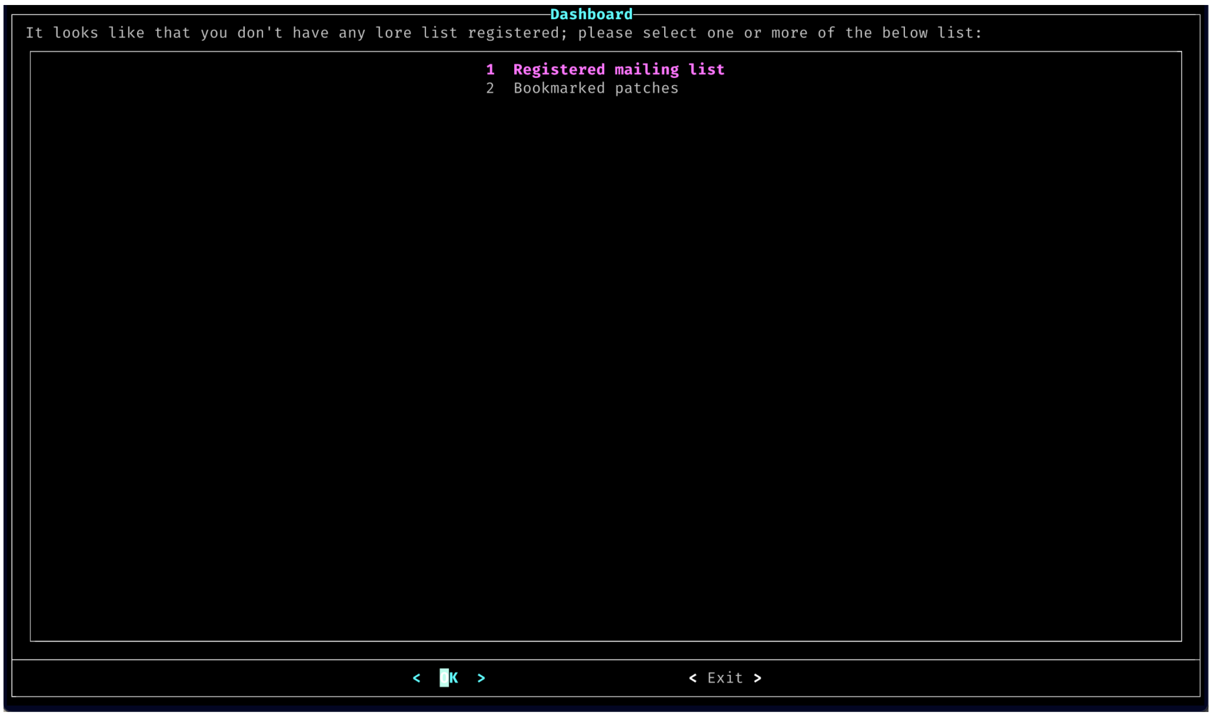
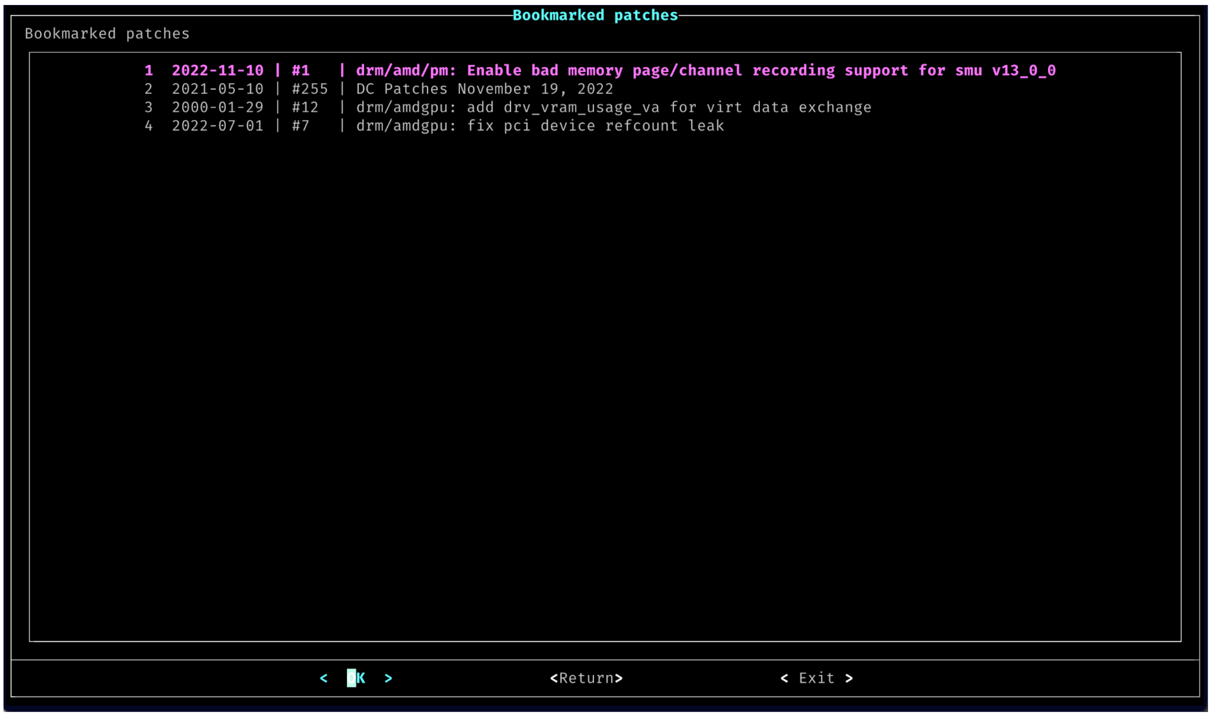

```
It looks like that you don't have any lore list registered; please select one or more of the below list:
```

Figure 2 - Dashboard (main screen) of the feature

As we can see, it lacks some important menus, like one to manage the registered mailing lists (only when first launching the feature can the user register/unregister lists) and another to configure the feature settings (preferred dialog theme, default local kernel source tree, etc.). Also, we can see that the message box (upper section of the screenshot) is not related to this screen.



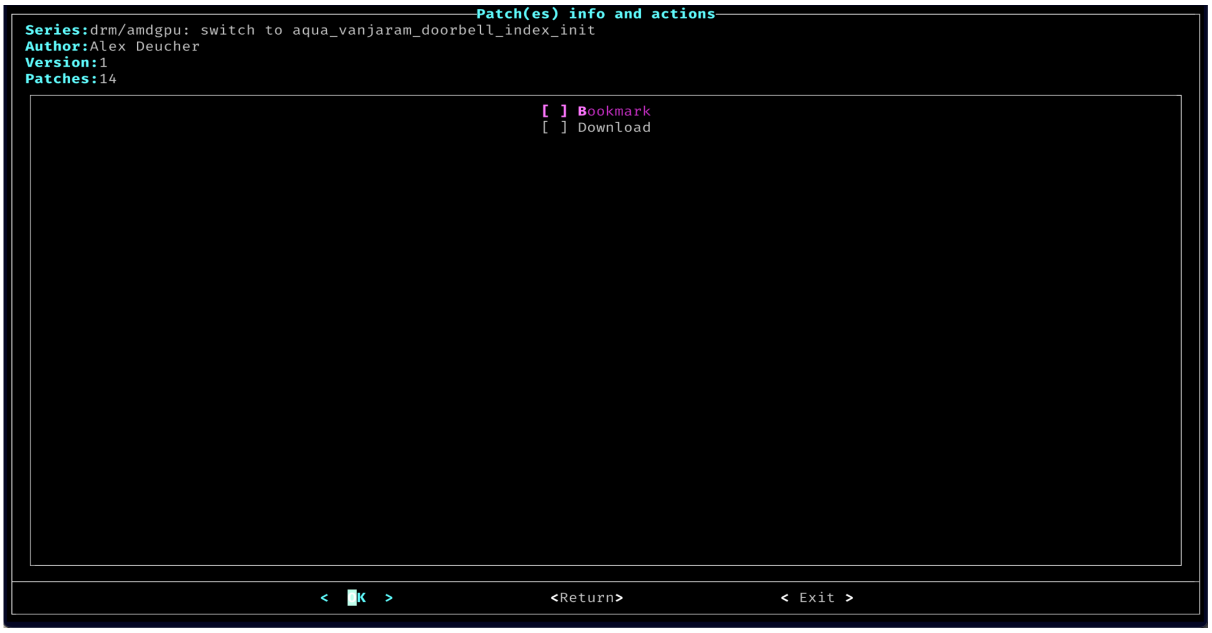
```
                                   Bookmarked patches
 Bookmarked patches

            1  2022-11-10 | #1   | drm/amd/pm: Enable bad memory page/channel recording support for smu v13_0_0
            2  2021-05-10 | #255 | DC Patches November 19, 2022
            3  2000-01-29 | #12  | drm/amdgpu: add drv_vram_usage_va for virt data exchange
            4  2022-07-01 | #7   | drm/amdgpu: fix pci device refcount leak

                      < OK >              <Return>              < Exit >
```

Figure 3 - Bookmarked Patches screen

The screenshot represents a desirable look and feel of the Bookmarked Patches screen. However, the patches displayed are hardcoded because the bookmarking of patches is not implemented.



```
                                Patch(es) info and actions
 Series:drm/amdgpu: switch to aqua_vanjaram_doorbell_index_init
 Author:Alex Deucher
 Version:1
 Patches:14

                                    [ ] Bookmark
                                    [ ] Download

                      < OK >              <Return>              < Exit >
```

Figure 4 - Patch (series) details and actions screen

There are many actions that we can add here. Aside from implementing the bookmark action, we could add 'Apply', 'Build', 'Deploy', 'Reply with Reviewed-by', 'Reply with Tested-by', 'View', and more. We could also refine the 'Download' action to let the user choose where to save the patch (series) and implement a way to schedule actions.

`kw upstream-patches-ui` interface mockup:

Below are some mockups of the main screens related to the feature.

- **Main screen of the feature (Dashboard):**

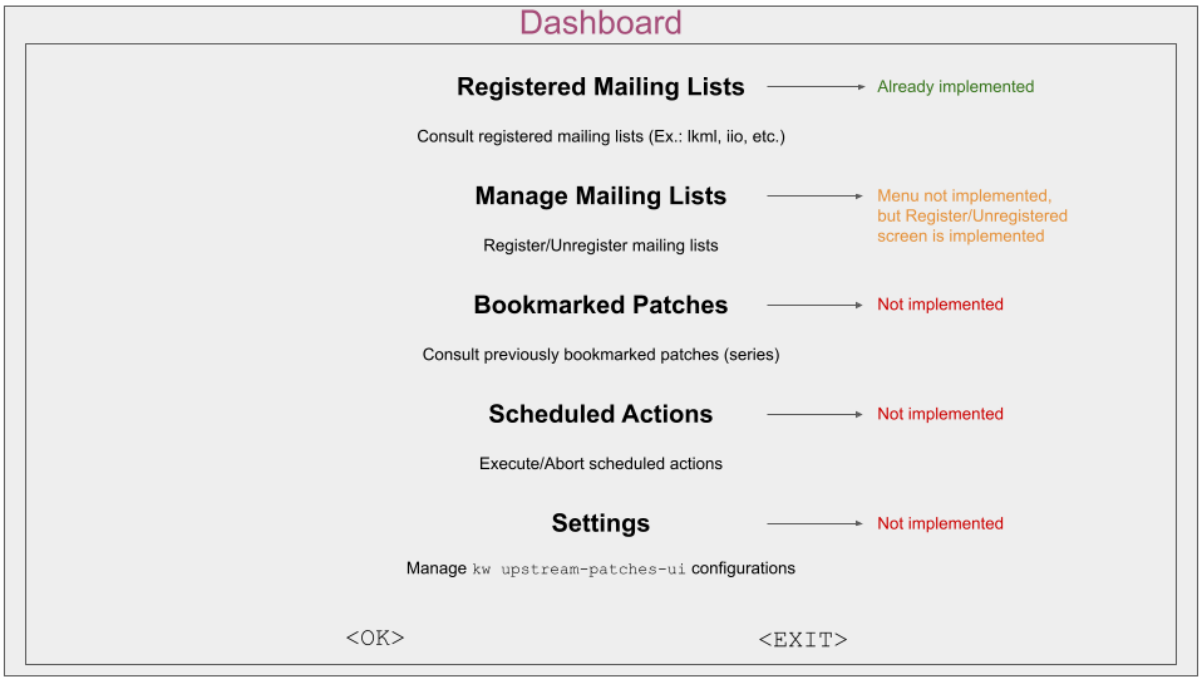
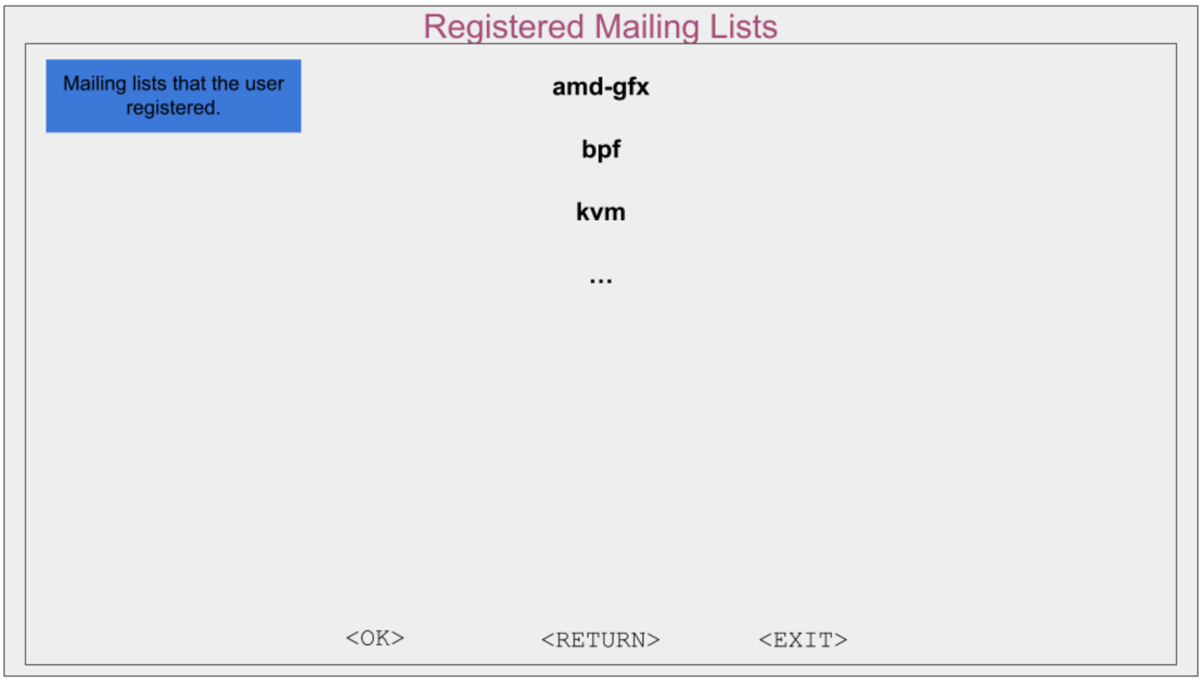

Dashboard
Registered Mailing Lists
Already implemented
Consult registered mailing lists (Ex.: lkml, iio, etc.)
Manage Mailing Lists
Menu not implemented, but Register/Unregistered screen is implemented
Register/Unregister mailing lists
Bookmarked Patches
Not implemented
Consult previously bookmarked patches (series)
Scheduled Actions
Not implemented
Execute/Abort scheduled actions
Settings
Not implemented
Manage kw upstream-patches-ui configurations
<OK>
<EXIT>

Figure 5 - Dashboard screen mockup

The Dashboard (**Figure 5)** is the first screen shown, besides when first launching the feature. Here the user will have access to the main menus:

1. Registered Mailing Lists: this screen is basically implemented and represents the menu to access the public mailing lists registered by the user.
2. Manage Mailing Lists: screen to register/unregister mailing lists. The screen itself is already implemented, however, there is no menu to access it from the Dashboard at the moment.
3. Bookmarked Patches: screen to view patches (series) marked previously by the user. For the same patch (series), the user will have almost the same view as if it was accessed through the 'Registered Mailing Lists' menu.
4. Scheduled Actions: screen with a list of patches (series) with actions scheduled to be executed in batch. Schedulable actions can be those that the user may want to run afterward that (potentially) can take some time, like applying a series of patches, building the series version of the kernel, deploying the series version of the kernel, etc.
5. Settings: screen to change configurations through the feature itself.

- **Registered Mailing Lists Sequence:**



Registered Mailing Lists

Mailing lists that the user registered.

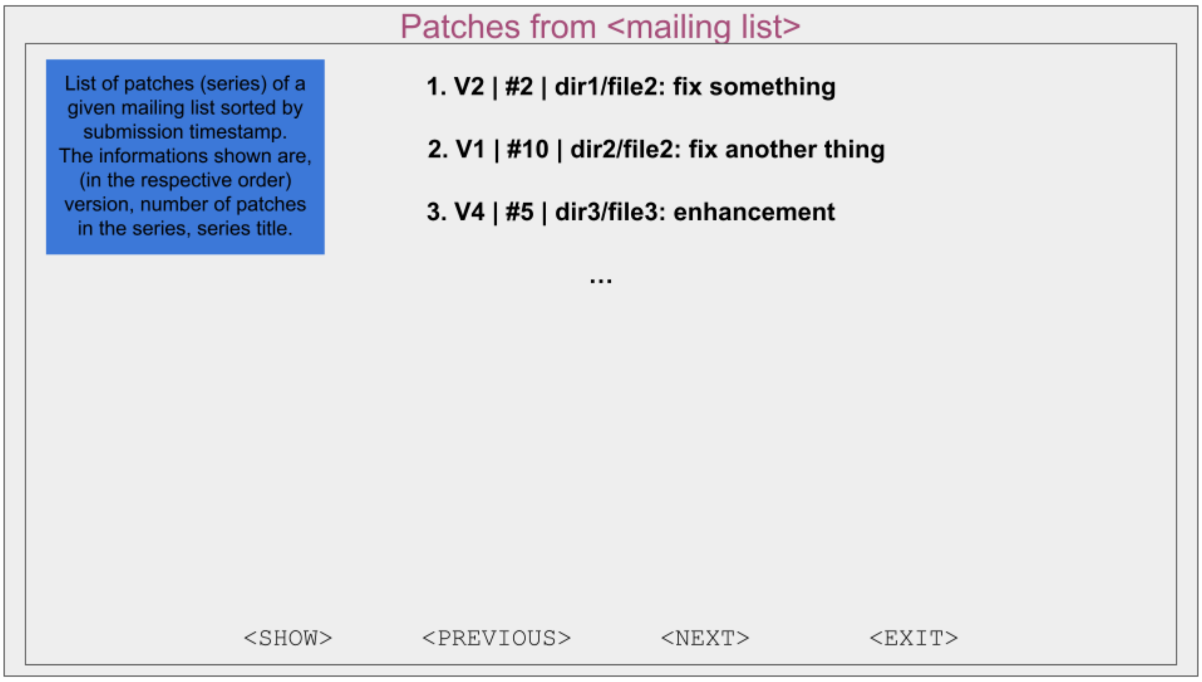
amd-gfx

bpf

kvm

...

Figure 6 - Registered Mailing Lists screen mockup



Patches from <mailing list>

List of patches (series) of a given mailing list sorted by submission timestamp. The informations shown are, (in the respective order) version, number of patches in the series, series title.

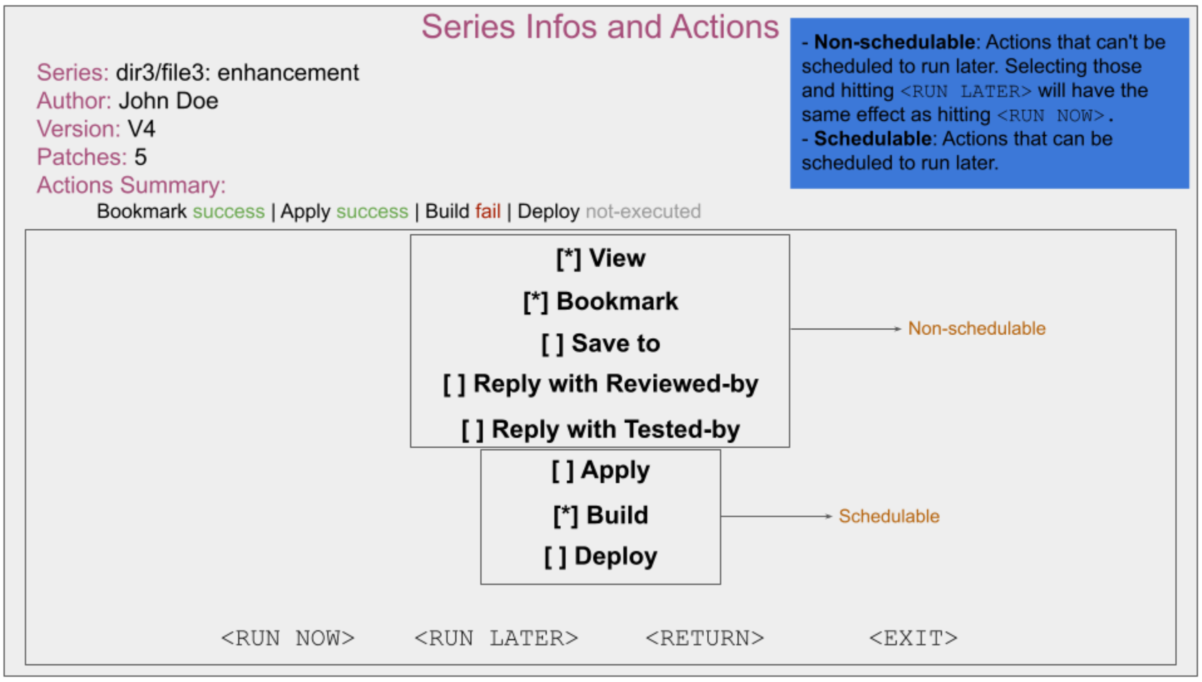
1. V2 | #2 | dir1/file2: fix something

2. V1 | #10 | dir2/file2: fix another thing

3. V4 | #5 | dir3/file3: enhancement

...

Figure 7 - Patches from given mailing list screen mockup


Series Infos and Actions

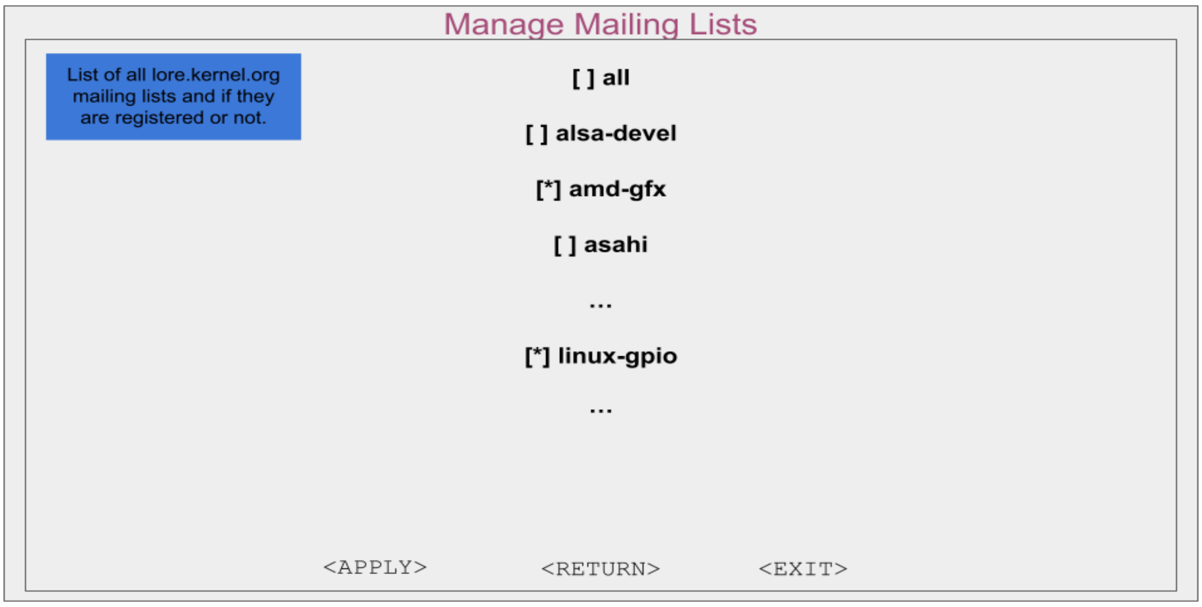
Series: dir3/file3: enhancement
Author: John Doe
Version: V4
Patches: 5
Actions Summary:
Bookmark success | Apply success | Build fail | Deploy not-executed

- **Non-schedulable**: Actions that can't be scheduled to run later. Selecting those and hitting `<RUN LATER>` will have the same effect as hitting `<RUN NOW>`.
- **Schedulable**: Actions that can be scheduled to run later.

Figure 8 - Series informations and actions screen mockup

A screen with the registered mailing lists will be displayed when selecting the Dashboard menu 'Registered Mailing Lists' (**Figure 6**). After the user chooses a mailing list, the series of the list will be displayed from the latest to the oldest (**Figure 7**). As the volume of series from a list can be enormous, the series will be displayed on pages, and the buttons <PREVIOUS> and <NEXT> can be used to navigate through them. By choosing a given series with the <SHOW> button, the user can consult general information on the top of the screen (series title, author, etc.), check the status of some actions (if they ran and if they were successful or not) and select actions to run. Note that there are two types of actions: ones that can't be run later (Non-schedulable) and those that can (Schedulable) (**Figure 8**). If the user chooses to run some actions later by hitting <RUN LATER>, those will be displayed in the 'Scheduled Actions' menu. Hitting <RUN LATER> with actions checked that are Non-schedulable will have the same effect as hitting <RUN NOW> for those.

- **Manage Mailing Lists Sequence**


Manage Mailing Lists

List of all lore.kernel.org mailing lists and if they are registered or not.

[ ] all

[ ] alsa-devel

[*] amd-gfx

[ ] asahi

...

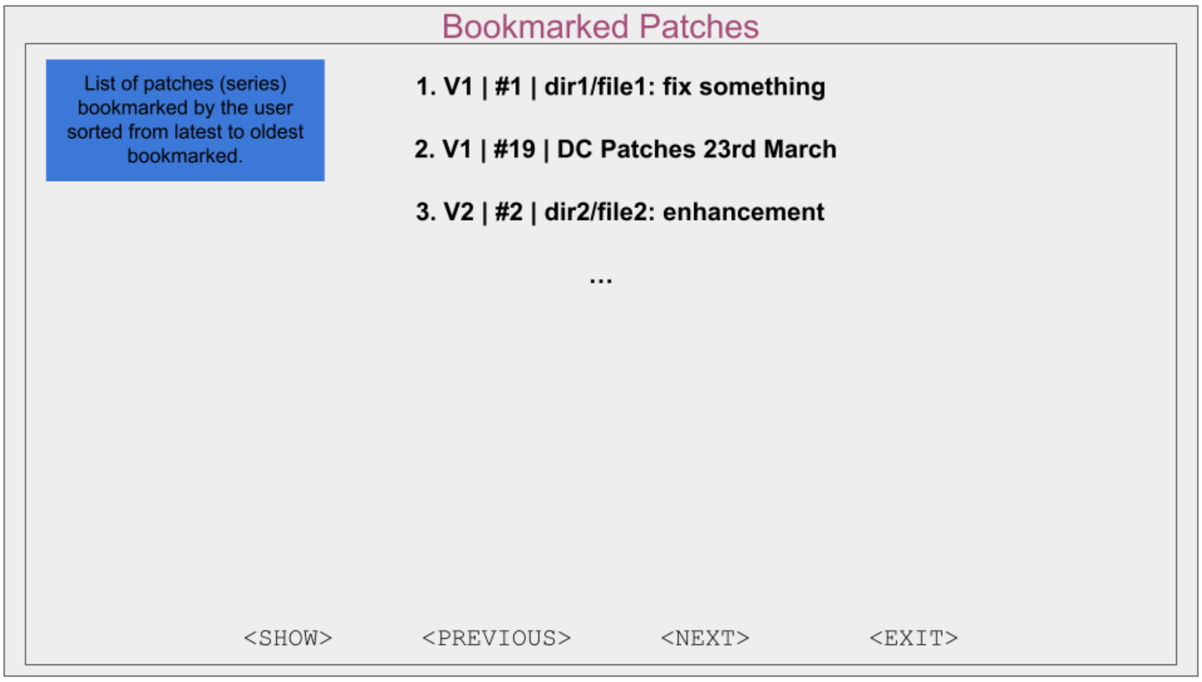
[*] linux-gpio

...

Figure 9 - Manage Mailing Lists screen mockup

By selecting the Dashboard menu 'Manage Mailing Lists' the screen to register/unregister mailing lists will be displayed (**Figure 9**). Important to note that the screen itself is already implemented.

- **Bookmarked Patches Sequence**


Bookmarked Patches

List of patches (series) bookmarked by the user sorted from latest to oldest bookmarked.

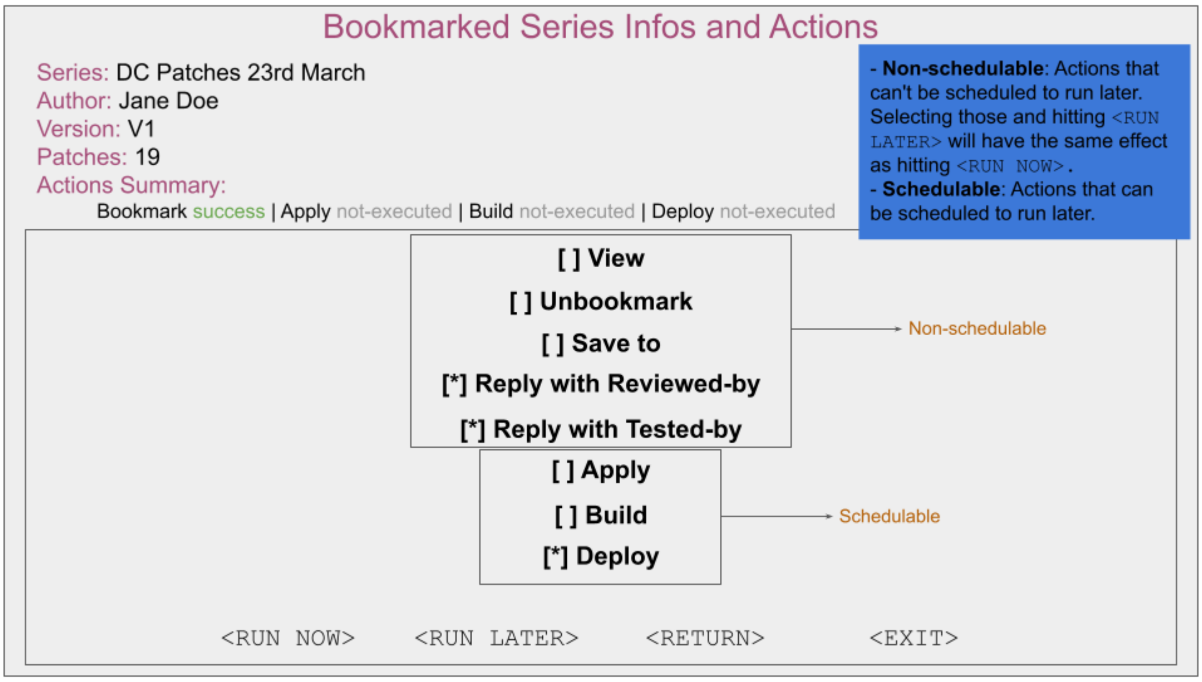
1. V1 | #1 | dir1/file1: fix something

2. V1 | #19 | DC Patches 23rd March

3. V2 | #2 | dir2/file2: enhancement

...

Figure 10 - Bookmarked Patches screen mockup


Bookmarked Series Infos and Actions

Series: DC Patches 23rd March
Author: Jane Doe
Version: V1
Patches: 19
Actions Summary:
Bookmark success | Apply not-executed | Build not-executed | Deploy not-executed

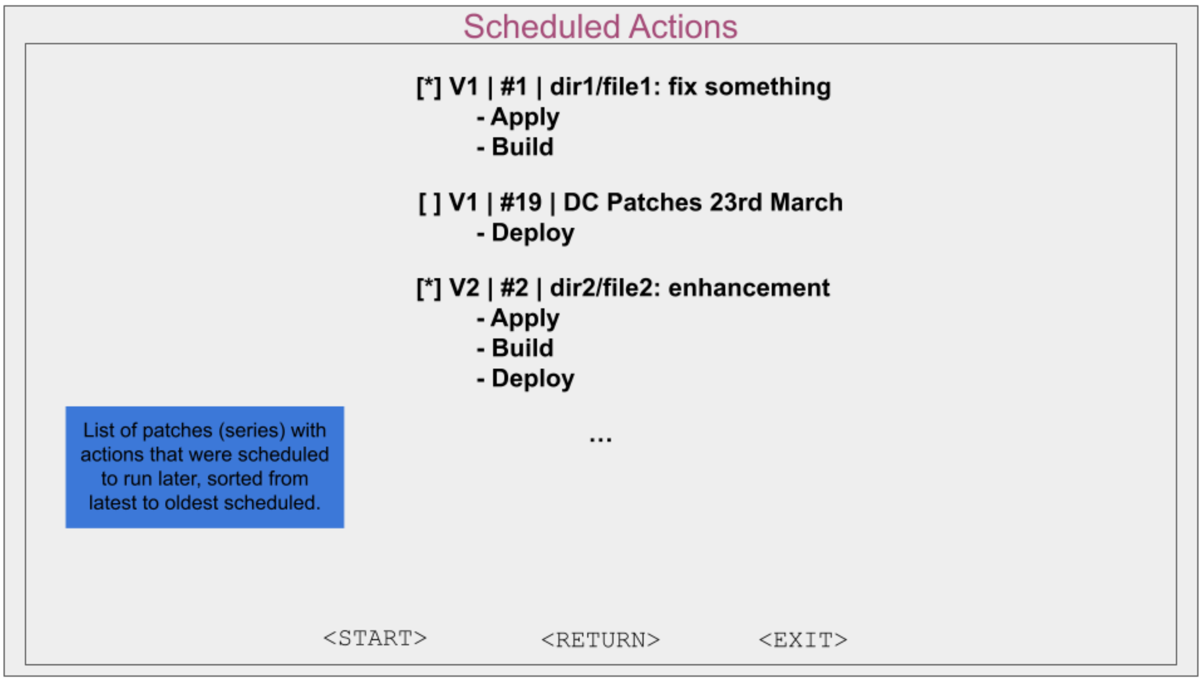
- **Non-schedulable**: Actions that can't be scheduled to run later. Selecting those and hitting `<RUN LATER>` will have the same effect as hitting `<RUN NOW>`.
- **Schedulable**: Actions that can be scheduled to run later.

[ ] View
[ ] Unbookmark
[ ] Save to
[*] Reply with Reviewed-by
[*] Reply with Tested-by
[ ] Apply
[ ] Build
[*] Deploy

Figure 11 - Bookmarked Series information and actions screen mockup

When accessing the Dashboard menu 'Bookmarked Patches', a screen with the list of the bookmarked series, from latest to oldest bookmark, will be displayed (**Figure 10**). By selecting a given series, a screen with its information and actions will be displayed (**Figure 11**). This screen is almost identical to the one displayed when selecting a series directly from a given mailing list (**Figure 8**), with the difference that it has an option 'Unbookmark' that removes the series from the bookmarks after it is run.

- Scheduled Actions Sequence


Scheduled Actions

[*] V1 | #1 | dir1/file1: fix something
- Apply
- Build

[ ] V1 | #19 | DC Patches 23rd March
- Deploy

[*] V2 | #2 | dir2/file2: enhancement
- Apply
- Build
- Deploy

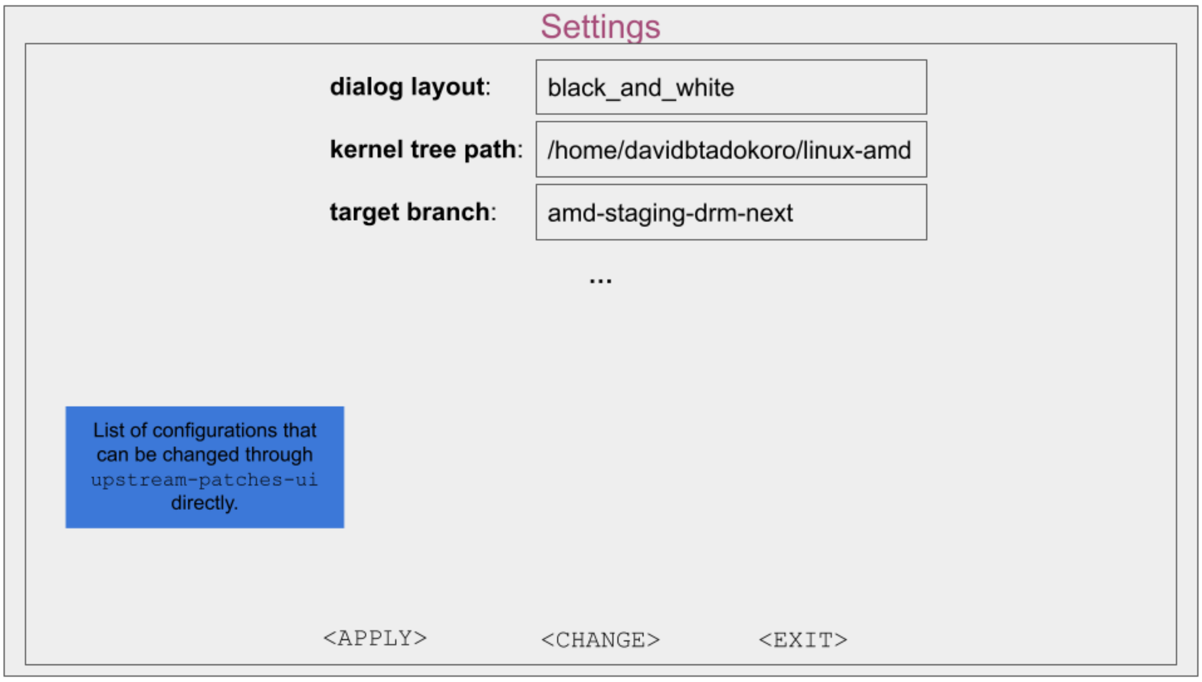
...

List of patches (series) with actions that were scheduled to run later, sorted from latest to oldest scheduled.

Figure 12 - Scheduled Actions screen mockup

By choosing the Dashboard menu 'Scheduled Actions', a screen with the list of series with actions that were scheduled from the 'Series Infos and Actions' (**Figure 8**) and 'Bookmarked Series Infos and Actions' (**Figure 11**), from the latest to oldest scheduled, will be displayed (**Figure 12**). By hitting `<START>`, the actions from the selected series will begin to run.

- **Settings Sequence**


Settings

dialog layout: black_and_white

kernel tree path: /home/davidbtadokoro/linux-amd

target branch: amd-staging-drm-next

...

List of configurations that can be changed through upstream-patches-ui directly.
<APPLY>
<CHANGE>
<EXIT>

Figure 13 - Settings screen mockup

When selecting the Dashboard menu 'Settings', a screen with all the configurations that can be set through `upstream-patches-ui` will be displayed (**Figure 13**). By hitting `<CHANGE>` with a given configuration selected, the user can edit the given configuration; and by hitting `<APPLY>`, the changes will be applied.

- **Notes**

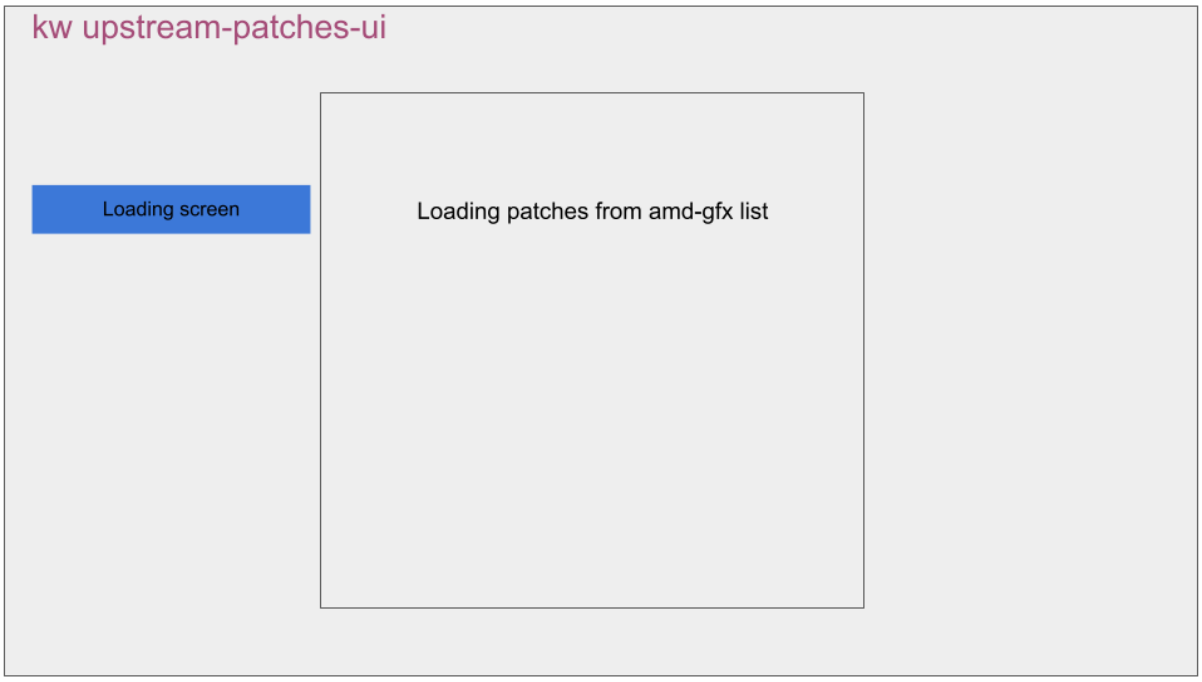


Figure 14 - Loading Screen Notification mockup

Selecting some menus (like selecting a given mailing list) and actions (like saving or building a series) will trigger a loading screen notification just to inform the user that there may be a delay between the request and the completion (**Figure 14**).

Wherever there is an `<EXIT>` button, hitting it will exit `upstream-patches-ui` and return to the shell. Also, wherever there is a `<RETURN>` button, hitting it will return to the previous screen. In the case of the `<PREVIOUS>` and `<NEXT>` buttons, hitting `<PREVIOUS>` on the first page will have the same effect as the `<RETURN>` button.

Deliverables

- `kw upstream-patches-ui`:
  1. Implement an interface that is user-friendly, simple, and without major bugs.
  2. Make the feature capable of downloading, building, and deploying patches and displaying the status of those actions to the user.
  3. Allow users to reply to the patches in the public mailing lists with Reviewed-by, Tested-by, and inline reviews.
- `kw mail`:
  1. Update feature codestyle.
  2. Fix known bugs.
  3. Improve feature where possible.

Proposal Timeline

Period of May 4 - May 28, 2023: Community Bonding

Week 1: May 4 - May 7

- Integrate `kw build` with `upstream-patches-ui`
- Add basic documentation for `upstream-patches-ui`
- Add Bash and Zsh completions for `upstream-patches-ui`

Week 2: May 8 - May 14

- Integrate `kw deploy` with `upstream-patches-ui`
- Get used to `kw mail` codebase

Week 3: May 15 - May 21

- Add 'Settings' menu to Dashboard screen
- Add `upstream-patches-ui` short option
- Map codestyling improvements to `kw mail`

Week 4: May 22 - May 28

- Add 'Manage Mailing Lists' menu to Dashboard screen
- `upstream-patches-ui` can't download patches with apostrophe in title
- Fix/update `kw mail` codestyle

Period of May 29 - July 9, 2023: Coding Phase 1

Weeks 5, 6, 7, 8 and 9: May 29 - July 2

- Refine the 'Download' action on `kw upstream-patches-ui`
- Make `upstream-patches-ui` query X patches
- Enable query for specific string from lore
- Add the possibility of reply a patch with Reviewed/Tested-by
- Add possibility of viewing patch changes through `upstream-patches-ui`
- Add inline review of patches
- Fix some bugs and make some enhancements to `kw mail`

Week 10: July 3 - July 9

- Refine a prototype of the feature that can be used for a full patch review routine

Period of July 10 - July 14, 2023: Midterm Evaluation

Midterm Evaluation week: July 10 - July 14

- Finalize and submit midterm evaluation

Period of July 10 - August 27, 2023: Coding Phase 2

Weeks 11, 12, 13 and 14: July 17 - August 6

- Refine `upstream-patches-ui` local database
- Revise `kw mail` documentation
- Experiment with other views for `kw upstream-patches-ui` besides `dialog`

Weeks 15, 16 and 17: August 7 - August 27

- Validate features and User Experience
- Make refinements and features improvements

Period of August 28 - September 4, 2023: Final Evaluation

Final Evaluation week: August 28 - September 4

- Mentors submit final GSoC contributors evaluations