surplus on wheels (s+ow)

surplus on wheels is a pure shell script to get your location using termux-location, process it through surplus, and send it to messaging service or wherever using “bridges”

surplus was made to emulate sending your location through the iOS Shortcuts app, and surplus on wheels complements it by running surplus automatically using a cron job. (but using it manually also works!)

installing

Important

s+ow is a Termux-first script, and will not work anywhere else unless you have a utility that emulates termux-location on `$PATH` alongside bridges that supports your platform

there are two notable ways to install s+ow:

1. as a standalone script
2. or, as a cron job

there is also an installation script for quickly getting started from a *fresh* termux installation

as a standalone script

1. firstly install python and termux-api if you haven't already:

   ```
   pkg install python termux-api
   ```

   also install the accompanying Termux:API app from F-Froid

2. install pipx if you haven't already:

   ```
   pip install pipx
   ```

3. install surplus:

   ```
   pipx install surplus
   ```

4. install surplus on wheels:

   ```
   mkdir -p ~/.local/bin/
   wget  -O ~/.local/bin/s+ow https://surplus.joshwel.co/spow.sh
   chmod +x ~/.local/bin/s+ow
   ```

> **Note**
>
> if `wget` throws a 404, see backup links

if `~/.local/bin` is not in your `$PATH`, add the following to your shell's rc file:

```
export PATH="$HOME/.local/bin:$PATH"
```

et voilà! s+ow is now setup. to actually send the message to a messaging platform, install an appropriate bridge

as a cron job

> **Important**
>
> these instructions rely on following the previous instructions

1. install necessary packages to run cron jobs:

   ```
   pkg install cronie termux-services
   ```

2. restart termux and start the cron service:

   ```
   sv-enable cron
   ```

3. set up the cron job:

   run the following command:

   ```
   crontab -e
   ```

   and add the following text:

   ```
   59 * * * *      bash -l -c "(SPOW_TARGETS="" SPOW_CRON=y s+ow)"
   ```

   > **Important**
   >
   > minimally fill in the `SPOW_TARGETS` variable before running s+ow. (see usage for more info)

   this will run s+ow every hour, a minute before the hour

   modify the variables as per your needs. see usage for more information

   et voilà! s+ow will now send a message every hour. feel free to experiment with the cron job to your liking. see crontab.guru if you’re new to cron jobs if you haven't already, install an appropriate bridge to actually send a message to a messaging platform

using installation scripts

> ⚠ **Warning**
>
> these scripts assume you're starting from a fresh base installation of Termux. if you have already cron jobs, then manually carry out the instructions in 'as a cron job'

> **Important**
>
> if not installed already, install Termux:API from F-Droid, **not the Play Store**

1. setup s+ow:

   ```
   wget -O- https://surplus.joshwel.co/termux.sh | sh
   ```

   > **Note**
   >
   > if `wget` throws a 404, see backup links

2. restart termux!
3. and finally, set up a cron job from step 3 onwards ('set up the cron job')

usage

environment variables

s+ow's behaviour can be customised environment variables, with `SURPLUS_CMD` being the only required variable:

1. `SPOW_TARGETS`
   a single line of comma-delimited chat IDs with bridge prefixes

   ```
   wa:000000000000000000@g.us,tg:-0000000000000000000,...
   ```

   in the example above, the WhatsApp chat ID is `wa:`-prefixed as recognised by the spow-whatsapp-bridge, and the Telegram chat ID is `tg:`-prefixed as recognised by the spow-telegram-bridge
2. `SPOW_CRON` (optional)
   set as non-empty to declare that s+ow is being run as a cron job

if running as a cron job, start s+ow one minute earlier than intended to account for the time it takes to run `termux-location` and `surplus`. s+ow assumes this and delays itself appropriately

setting it to `n` will also be treated as if it were empty

3. `SPOW_PRIVATE` (optional)
   set as non-empty to discard all logs when s+ow is done:

   - `$HOME/.cache/s+ow/out.log` will be set to `/dev/null`
   - `$HOME/.cache/s+ow/err.log` will be set to `/dev/null`
   - `$HOME/.cache/s+ow/location.net.json` will be cleared after use locating the device
   - `$HOME/.cache/s+ow/location.gps.json` will be cleared after use locating the device
   - `$HOME/.cache/s+ow/location.json` will be cleared after use locating the device
   - `$HOME/.cache/s+ow/surplus.out.log` will be cleared after use generating the message
   - `$HOME/.cache/s+ow/surplus.err.log` will be set to `/dev/null`
   - `$HOME/.cache/s+ow/message` will be cleared after all bridges has sent the message

   > ⚠ **Warning**
   >
   > the only file not cleared is s+ow's last successful message file, `$HOME/.cache/s+ow/last`, as s+ow uses this as the first fallback message if it couldn't locate the device in time. if you're fine with using the `LOCATION_FALLBACK` string, feel free to modify your cron job to remove this file after running s+ow

   setting it to `n` will also be treated as if it were empty

4. `SURPLUS_CMD` (optional)
   the custom invocation used when calling surplus, modify this if you want to add certain flags

   this defaults to `surplus -td`

   > ⚠ **Warning**
   >
   > when overriding, ensure you also have `-td` (`--using-termux-location` and `--debug`) in your custom invocation!

5. `LOCATION_CMD` (optional)
   the custom invocation used when calling `termux-location`, modify this if you want to bodge together surplus on wheels on non-termux systems. see (emulating `termux-location`) for more information

   this defaults to `termux-location`

6. `LOCATION_PRIORITISE_NETWORK` (optional)
   set as non-empty to declare that s+ow can just use network location instead of GPS if GPS is taking too long.
   you should only turn this on if punctuality means that much to you, or you're in a country with cell towers close by or everywhere, like Singapore

   the JIDs can be obtained by sending a message to the user/group, while running `s+ow mdtest`, and examining the output for your message. JIDs are email address-like strings setting it to `n` will also be treated as if it were empty

7. `LOCATION_TIMEOUT` (optional)
   set as a number to override the default first location timeout of `50`
8. `LOCATION_FALLBACK` (optional)
   a string that can be formatted with three numbers using `%d`:

   a. s+ow's status

   b. number of location attempts before giving up

   c. type of message sent

   see details on notification numbers for the meanings of each number. 'a', 'b' and 'c' map to `A`, `B` and `C`

   defaults to `%d%d%d?`

faking locations

> sometimes you gotta do what you gotta do

you can fake your s+ow messages by either:

1. setting a dummy `last` file in s+ow cache

   `$HOME/.cache/s+ow/last` is used as the fallback response when a part of s+ow (either `termux-location` or `surplus` errors out). you can set this file to whatever you want and just turn off location on your device
2. setting a `fake` file in s+ow cache

   Warning

   s+ow uses the `read` command to read the file. as such, it is possible for s+ow to prematurely stop reading the file if the file does not contain a trailing newline.

   you can also write text to `$HOME/.cache/s+ow/fake` to fake upcoming messages. the file is delimited by empty lines. as such, arrange the file like so:

```
The Clementi Mall
3155 Commonwealth Avenue West
Westpeak Terrace
129588
Southwest, Singapore

Westgate
3 Gateway Drive
Jurong East
608532
Southwest, Singapore

...
```

on every run of s+ow, the first group of lines will be consumed, and the file will be updated with the remaining lines. if the file is empty, it will be deleted

details on notification numbers

after each run, or if s+ow had to use a location fallback string, s+ow notifies you:

> **surplus on wheels**
>
> Run has finished.
>
> Singapore Conference Hall
> 7 Shenton Way
> 068809
> Central, Singapore
>
> (A, B, C, D)
> [lc:W sp:X sm:Y - Z]

> **surplus on wheels has errored**
>
> (A, B, C, D)
> [lc:W sp:X sm:Y - Z]

the top line denotes general statuses:

- `A`: s+ow's status
  - `0` is nominal
  - `1` is a termux-location error
  - `2` is a surplus error
  - `3` is a bridge/message send error
- `B`: number of location attempts before giving up
- `C`: type of message sent
  - `0` for freshly made sharetext
  - `1` for recycling a previous successful location sharetext (`last` file)
  - `2` for using fallback template
- `D`: number of bridge failures
- `E`: each bridge's return code

the bottom line details on how long s+ow spent on each stage:

- `W`: time to locate
- `X`: time to run surplus

- `Y`: time to send message(s)
- `Z`: total run time

help! a bridge isn't working!

cool. do the following:

1. log out and log back in and try again
2. if that didn't fix it, update/reinstall the bridge and try again
3. run the bridge's executable directly to see if there's any connection issues
   look at your bridge's installation instructions to find out where it's located at. or, use the `which` command
4. if it connected successfully, or you see no errors, try typing in one of the targets you've set in `SPOW_TARGETS` for the bridge, and then press the enter/return key

✕ **Failure**

on the off chance you reinstalled the bridge, and it still failed either step 3 or 4, the bridge itself is faulty. file a bug report/issue with the bridge's project page or maintainer and tell them where it failed (was it connecting to the messaging service? or failure to send a message?)