Утилита ilogger

1 Описание

Консольная утилита ilogger перехватывает управляющие действия пользователя и сохраняет их в stdout.

Под «управляющими действиями» понимаются следующие события:

- нажатие/отпускание клавиши на клавиатуре,
- нажатие/отпускание левой/средней/правой кнопки мыши,
- вращение колеса мыши,
- перемещение указателя мыши по экрану.

Каждое действие сохраняется в stdout как отдельная запись, которая выводится в текстовом или бинарном формате (в зависимости от значения опции `--ioformat`, см ниже).

Одновременно утилита читает свой stdin. Каждая запись из стандартного ввода также являет собой «действие пользователя» в таком же формате. Утилита имитирует эти действия (в дальнейшем называемые «собственными»), как будто их совершил пользователь.

Собственные действия также перехватываются и логгируются или не лоогируются в stdout (в зависимости от значения опции `--ownaction`, см ниже)

2 Записи ввода/вывода

2.1 События клавиатуры

2.1.1 Текстовый формат

Нажатию/отпусканию клавиши соответствуют строки:

```
Key press: n
Key release: n
```

где n – число идентифицирующее нажатую/отпущенную клавишу.

2.1.2 Бинарный формат

'ilog' – сигнатура, 4бт

00/01 — действие пользователя/собственное, 1бт

01 — клавиатура, 1бт

00/01 — press/release, 1бт

n – номер клавиши, 2бт

2.2 Кнопки мыши

2.2.1 Текстовый формат

Нажатию/отпусканию кнопки мыши соответствуют строки:

```
Mouse button press: LEFT
Mouse button press: MIDDLE
Mouse button press: RIGHT
Mouse button release: LEFT
Mouse button release: MIDDLE
Mouse button release: RIGHT
```

2.2.2 Бинарный формат

'ilog' – сигнатура, 4бт

00/01 — действие пользователя/собственное, 1бт

00 — мышь, 1бт

00/01 — press/release, 1бт

01/02/03 — кнопка LEFT/MIDDLE/RIGHT, 1бт

2.3 Колесо мыши

2.3.1 Текстовый формат

Вращению колеса мыши соответствуют строки:

```
Mouse wheel: UP
Mouse wheel: DOWN
```

2.3.2 Бинарный формат

'ilog' – сигнатура, 4бт

00/01 — действие пользователя/собственное, 1бт

00 — мышь, 1бт

02 — wheel, 1бт

00/01 – down/up, 1бт

2.4 Перемещение указателя мыши

2.4.1Текстовый формат

Перемещению указателя мыши соответствуют строки:

```
Mouse move: X=nx, Y=ny
где nx, ny -координаты указателя в пикселях.
```

2.4.2 Бинарный формат

'ilog' – сигнатура, 4бт

00/01 — действие пользователя/собственное, 1бт

00 — мышь, 1бт

03 — move, 1бт

nx – x координата, 4бт

ny – y координата, 4бт

3 Опции утилиты

Утилита воспринимает следующие опции:

- ***--skip n***
- ***--ioformat <format>***
- ***--ownaction <method>***

3.1 Опция --skip n

Определяет паузы между выводами утилиты. После перехвата (и вывода в stdout) очередного события, утилита отчитывает «молчаливый период» продолжительностью n миллисекунд. В течении этого периода события (как пользователя, так и «собственные») перехватываются, но в sdtdout не выводятся.

По истечение молчаливого периода, в stdout выводится 1 событие и начинается новый период.

Допустимые значения: 0 – 1000, умолчание: 0.

3.2 Опция --ioformat <format>

Определяет формат ввода/вывода (см выше).

Допустимые значения:

`normal` – текстовый формат (умолчание)

`binary` – бинарный формат.

3.3 Опция --ownaction <method>

Определяет реакцию утилиты на «собственные» события (см. выше).

Допустимые значения:

`normal` – вывод собственных событий никак не отличается от событий пользователя (умолчание).

`skip` – собственные события не выводятся в stdout.

`highlight` – в текстовом формате собственные события выводятся в stdout с префиксом '==>', например

```
==>Key press: 116
==>Key release: 116
```

В бинарном формате 1й байт после сигнатуры (источник события) устанавливается в 01 (см выше).

4 Сборка утилиты из исходников

4.1 Платформы

Утилита может быть собрана и исполнена на следующих платформах:

1. Windows: автор использовал Windows 11, но, видимо, все будет работать и на других (не слишком древних) версиях.
2. Linux: автор использовал Ubuntu 22.04, но, видимо, все будет работать и на других (не слишком древних) версиях.
3. MacOS: автор использовал MacOS Ventura 13.0 , но, видимо, все будет работать и на других (не слишком древних) версиях.

4.2 Компилятор

Утилита написана на C++. В исходных текстах использованы некоторые средства (thread, chrono) из Стандарта C++11. Соответственно, компилятор должен поддерживать -std=c++11 или более позднюю версию.

4.3 Windows

Для сборки под Windows необходимо:

1. Установить git, https://git-scm.com/download/win
2. Настроить специфичную для Винды обработку строк в git:

   ```
   git config --global core.autocrlf true
   ```

   Мотивация: https://tokmakov.msk.ru/blog/item/710
3. Установить MinGW-w64, http://winlibs.com/
4. Скачать исходники:

   ```
   git clone https://github.com/ao-kulpin/ilogger.git
   ```
5. Зайти (с помощью cd) в папку с исходниками и исполнить команду:

```
mingw32-make windows
```

В результате будет построен исполняемый файл `ilogger.exe`.

4.4 Linux

Для сборки под Linux необходимо:

1. Установить git для вашей версии ОС. Например, для Ubuntu:

```
sudo apt update
sudo apt install git
```

2. Установить компилятор g++, если его нет в системе (как правило он уже предустановлен).
3. Установить библиотеку libx11, https://launchpad.net/ubuntu/+source/libx11:

```
sudo apt update
sudo apt-get install libx11-dev
```

4. Установить библиотеку libxtst, https://launchpad.net/ubuntu/+source/libxtst:

```
sudo apt update
sudo apt -y install libxtst-dev
```

5. Скачать исходники:

```
git clone https://github.com/ao-kulpin/ilogger.git
```

6. Зайти (с помощью cd) в папку с исходниками и исполнить команду:

```
make linux
```

В результате будет построен исполняемый файл `ilogger`.

4.5 MacOS

Для сборки на MacOS необходимо:

1. Установить пакет Xcode Command Line Tools, (см https://www.freecodecamp.org/news/install-xcode-command-line-tools/). Этот пакет содержит все инструменты, необходимые для скачивания (git) и компиляции (make, g++) исходников.
2. Скачать исходники:

```
git clone https://github.com/ao-kulpin/ilogger.git
```

3. Зайти (с помощью cd) в папку с исходниками и исполнить команду:

```
make macos
```

В результате будет построен исполняемый файл `ilogger.app`.

5 Исполнение утилиты

5.1 Linux

Для перехвата и генерации системных событий ilogger использует стандартные средства X

Window сервера:

https://ru.wikipedia.org/wiki/X_Window_System

который отвечает за взаимодействие с пользователем (в частности отрисовку окон) во всех версиях Линукс. Однако в последних версиях системы (в частности Убунту 22.04) по умолчанию включена подсистема Wayland:

https://ru.wikipedia.org/wiki/Wayland

которая «оптимизирует» базовые операции X Window. С включенной Wayland в некоторых приложениях/окнах события не перехватываются. В частности это происходит в term (терминал), в функционально эквивалентном xterm – все перехатывается.

Отключение Wayland является легитимной операцией. В Убунту 22.04 отключение сводится к раскомментариванию одной строки в `/etc/gdm3/custom.conf`:

```
[daemon]
     # Uncomment the line below to force the login screen to use Xorg
     #WaylandEnable=false
```

Без Wayland, ilogger перехватывает события во всех окнах/приложениях.

5.2 MacOS

Перехват и генерация системных событий считается в MacOS “опасной” операцией, которая требует специальных разрешений. В отсутствие оных ilogger выдает сообщение

```
Failed to create event tap
```

и аварийно завершается.

Настройка разрешений - это довольно мутная тема. Следование рекомендациям официальной документации к успеху не привела.

Могу лишь поделиться своим личным опытом, который сработал моей на конкретной ОС (`MacOS Ventura 13.0`):

Зайти в меню системных настроек:

```
<Apple> -> <System Settings> -> <Privacy & Security> -> <Accessibility>
```

Откроется список приложений, которым “разрешено управлять этим компьютером”. В этот список я добавил приложение Terminal. После этого ilogger стал успешно запускаться из терминала.

Добавление в этот список самого приложения ilogger.app положительного эффекта не дает.

Мутная тема, с которой я до конца не разобрался…