Linux Speakup Makes Linux Talk to Users: Past and Future

Okash Khawaja / Contributor to Speakup

@binarydebt
What we will cover

• What is speakup
• Past, present and future
• Some technical details:
  • overall arch
  • hardware synths
  • ttyio architecture
• Development process
• Q&A
What is speakup?

- Piece of software that reads out what's on console and what you type with keyboard
  - Two modes:
    - Read text printed by applications
    - Whole set of keyboard shortcuts to control the output, including going back and reading what was previously printed by application and still on screen
- Linux only
- Entirely kernel-based, although exposes interface to user space
- Starts at early boot and lasts until shutdown
History

- Originally written by a blind person
- Started by Kirk Reiser in 1998
- Merged into Linux staging directory in late 2010
- Now work is underway to move it out of staging into kernel proper
- External device or ISA card for additional serial port
• tty based vs directly accessing ports
  • unique design because it’s purely a software driver and needs to operate on serial port
• console content and keyboard snooping done properly
  • register_vt_notifier()
  • register_keyboard_notifier()
• proper unicode support added: all languages can be supported seamlessly
Future

- Move from staging to mainline
- Fix bugs: garbled text
- New features: usb drivers, auto load, internationalisation of speakup messages
- More here: https://github.com/bytefire/speakup/issues
Overall Architecture

- core (speakup.ko)
- synths (speakup_*.ko)
- flow of data: {console, keyboard} -> notifiers -> circular buffer -> kthread -> synths (serial port or software interface)
Core

- main.c contains speakup core
- main.c + supporting files yield speakup.ko
- upon loading:
  - perform some initialisations (spk_vars, add virtual keyboard, initialise an internal representation of every console etc)
  - initialise sysfs interface
  - register tty ldisc
  - synth_init()
  - register keyboard notifier
  - register vt notifier
  - speakup kthread (take from buffer and push to serial or software interface)
Device Architecture

- Two types of synths: hardware and software
- Inside hardware, two types of devices: internal ISA cards (old) and external
- We will focus on external hardware synths and software synths
Device Architecture – Hardware Synths
Different synth chipsets: DoubleTalk, Apollo, DecTalk, Braille ‘n Speak and more

Connect using serial protocol – either serial port RS-232 or serial over USB

Pure USB support planned
Device Architecture – Support from core

struct spk_synth {
    ...
    struct spk_io_ops *io_ops;
    int (*probe)(struct spk_synth *synth);
    void (*release)(void);
    const char *(*synth_immediate)(struct spk_synth *synth,
    const char *buff);
    void (*catch_up)(struct spk_synth *synth);
    void (*flush)(struct spk_synth *synth);
    int (*is_alive)(struct spk_synth *synth);
    void (*read_buff_add)(u_char);
    unsigned char (*get_index)(struct spk_synth *synth);
    ...
};
Device Architecture – CPU to Synth

- TTY layer to send data to device
- struct spk_io_ops {
  int (*synth_out)(struct spk_synth *synth, const char ch);
  int (*synth_out_unicode)(struct spk_synth *synth, u16 ch);
  void (*send_xchar)(char ch);
  void (*tiocmset)(unsigned int set, unsigned int clear);
  unsigned char (*synth_in)(void);
  unsigned char (*synth_in_nowait)(void);
  void (*flush_buffer)(void);
};
Device Architecture – Synth to CPU

- Why synth-to-CPU traffic? Synths report their speaking progress to CPU.
- Use line discipline to communicate from device to CPU
- include/linux/tty_ldisc.h:struct tty_ldisc_ops
- Methods we care about are:
  - open()
  - close()
  - receive_buf2()
Speaking from early boot

• How it worked before: inb(), outb() etc
• For TTY, handle to tty_struct is needed
• Usually obtained from user space by opening /dev/ttyUSB0
• But we want to start speaking before there is file system or even user space
Speaking from early boot

• Need to open tty from kernel space
• Introduced tty_kopen() and tty_kclose()
• Needed addressing potential synchronisation issues with user-space open of same tty
• Affected user-kernel interface because an open() of a tty could now return -EBUSY
Device Architecture – Putting it together

- Console
  - struct vc_data
  - vt_notifier_call()

- Keyboard
  - keyboard_notifier_call()

- buffers.c
  - thread.c
  - kthread
  - CALLBACK: spk_synth->catch_up()

- Device Modules

- TTYIO
  - TTY Layer
    - ldisc
    - tty_struct
  - Serial Port
    - Hardware Synth

- SERIALIO (Legacy)
  - Serial Port
    - Hardware Synth

- speakup_soft.ko
  - .read()
  - .write()

- /dev/softsynth
Software Synths

- Two software synths for different motivations:
  - speakup_soft.ko
  - speakup_dummy.ko
Software Synths: speakup_soft.ko

- Like any other synth but doesn’t push speech to device
- Instead exposes an interface to user space
- Misc device /dev/softsynth that can be read by a user space program in the same way as hardware synth would read and then speak.

```
speakup.ko ↔ speakup_soft.ko ↔ /dev/softsynth ↔ espeakup ↔ espeak → physical speakers
```

- Limitations: requires user space to be up before it can start speaking and doesn’t speak all shutdown messages
Software Synths: speakup_dummy.ko

- Useful for development purposes
- Typical development set up with speakup_dummy:
  - qemu with serial port tied to a file on host
  - load speakup_dummy with dev param pointing to the serial port
- In essence:
  - host$ qemu -serial stdio
  - guest# modprobe speakup_dummy
Software Synths: speakup_dummy.ko

- Get all output in text
  - E.g. useful to debug garbled speech
- No extra layers (e.g. user space components) involved
Development Process

• Like general kernel development process but simpler and friendlier :)  
• Internal reviews! thanks to Samuel  
• Helpful community:
  • Testing  
  • Help with getting hardware devices  
  • Tech savvy
• qemu: compile on host or inside qemu VM
• speakup_dummy for most cases
• hardware synth for hardware features / debugging
• careful to load correct driver for the synth
  • speakup_apollo.ko on double talk
  • "just goes to show you when you give double talk to a god" - one of the mailing list replies
Development Process (contd.)

- great way to learn
- real community impact
- we're working towards moving it out of staging into main kernel
- plenty of interesting work ahead:
  - Complete transition out of staging
  - Support for USB synths
  - USB autoload
  - Bug fixes – possible races
  - Internationalisation of speakup messages
  - See https://github.com/bytefire/speakup/issues for more
Q&A