



Unix-Features: Überblick (1)

- Wie alle (modernen) BS bietet Unix eine Abstraktion der Hardware und ein Interface, über das Prozesse diese nutzen können
- Prozessor (Multi-Tasking, Scheduling)
- Prozessverwaltung (`fork`, `exec`, `wait`, `exit`)
- Festplatte/Diskette (Dateisystem)
- Prozess-Kommunikation und -Synchronisation
- Netzwerk: TCP/IP, Sockets
- Multi-User-Betrieb

Prozessor (Scheduling)

- Unix stellt die Ressource CPU mehreren Prozessen zur Verfügung
- je nach Unix-Version verschiedene Scheduling-Strategien
- Priorisierung von Prozessen (`nice` values), Steuerung über `nice()` und `setpriority()`
- kein Prozess kann die CPU monopolisieren

Prozessverwaltung (1)

- Prozesse bei Unix in Baumstruktur organisiert („Vater/Sohn“)
- Neue Prozesse werden mit `fork()` als ident. Kopie des aufrufenden Prozesses erzeugt
- Prozess kann mit `exec()` ein anderes Programm nachladen
- `exit()` beendet Prozess (mit Rückgabewert)
- `wait()`, `waitpid()` warten auf Ende eines Kindprozesses, Auswerten des Rückgabewerts

Prozessverwaltung (2)

- Intern verwaltet Unix Prozesse über Prozess-Kontroll-Blöcke (PCBs)
- jeder Prozess hat eigenen Adressraum (Speicherschutz), der zudem vom Adressraum des Unix-Kernels getrennt ist
- traditionell: keine Threads (nur über User-Level-Bibliothek)
- Context Switch wechselt von einem Prozess zum nächsten (Scheduler entscheidet, wann und zu welchem Prozess)

Dateisystem (2)

- System Calls für Zugriff auf Dateien
 - `fd=open()` öffnet Datei, gibt file descriptor zurück
 - `read(fd, ...)` liest aus Datei
 - `write(fd, ...)` schreibt in Datei
 - `lseek(fd, ...)` springt zu Position in Datei
 - `close(fd)` schließt Datei
- Interne Umsetzung oft über Virtual Filesystem (VFS)
- unterstützt mehrere FS und diverse Hardware

Dateisystem (1)

- Alle Unix-Dateisysteme haben gemeinsame Eigenschaften
 - hierarchisches Dateisystem (Unterordner)
 - Medien über Mountpoints in Baum integriert
 - Inodes verwalten Eigenschaften und Blockliste einer Datei
 - Inodes enthalten *keinen* Dateinamen
 - Verzeichnisse sind spezielle Dateien mit Zuordnungen
Dateiname → Inode-Nummer
 - freie Datenblöcke und freie Inodes in Bitmaps verwaltet
 - Superblock enthält Verwaltungsinformationen für gesamtes Dateisystem

IPC, Prozess-Synchronisation

- Signal-Mechanismus
 - Signal-Handler, `signal()`
 - Signal senden, `kill()`
- Auch BS selbst verwendet Signale
- Synchronisation über Mutexe und/oder Semaphore

Netzwerk

- TCP/IP (Verbindungen), UDP (verbindungslos)
- Sockets (Client/Server)
- Verwendung von Sockets ähnlich wie Zugriff auf Dateien,
 - `socket()`,
TCP: `SOCK_STREAM`,
UDP: `SOCK_DGRAM`
 - I/O multiplexing mit `select()`

ULIX: Features, Einschränkungen (1)

ULIX 0.13 ist erste veröffentlichte Release (Sept. 2015).

Im Kurs: Ulix 0.08 und Mini-Versionen

- **Prozesse:**
 - ✓ `fork`, `exit`, `wait`, `exec` funktionieren;
RR-Scheduler
 - ✗ aktuelle Version ohne Prioritäten (→ BA, fertig)
- **Dateisystem:**
 - ✓ VFS, Hardware: Disketten, IDE-Platten, Minix-v2-FS
 - ✗ `mkdir`, `rmdir` fehlen in Version 0.08,
Symlinks nur teilweise unterstützt

Multi-User-Betrieb

- Unix kennt mehrere Benutzer, `root` (`uid=0`) ist Systemverwalter
- Anmeldung über Username, Passwort
- Benutzergruppen
- Dateizugriff: Zugriffsrechte (für Besitzer, Gruppe, sonstige)
- Signalversand: nur an eigene Prozesse

ULIX: Features, Einschränkungen (2)

- **Signale:**
 - ✓ Datenstrukturen im PCB vorhanden, `signal()` trägt Handler ein, `kill()` verschickt Signal
 - ✗ empfangene Signale werden ignoriert (außer `SIGKILL`)
- **Speicher:**
 - ✓ Paging: Seitentabellen für jeden Prozess, Kernel-Adressraum wird beim Wechsel in Kernel-Mode sichtbar
 - ✗ Auslagern / Wiedereinlagern von Seiten (ab V. 0.12)

ULIX: Features, Einschränkungen (3)

- **System Calls:**
 - ✓ System Call Interface, erlaubt Registrieren neuer Syscall-Handler.
Standard-Syscalls für alle Kernel-Funktionen vorhanden
User-Mode-Bibliothek enthält Funktionen, welche diese Syscalls aufrufen
- **Interrupts:**
 - ✓ Interrupt-Handler für Timer, FDC, IDE, Tastatur, serielle Schnittstelle

ULIX: Features, Einschränkungen (5)

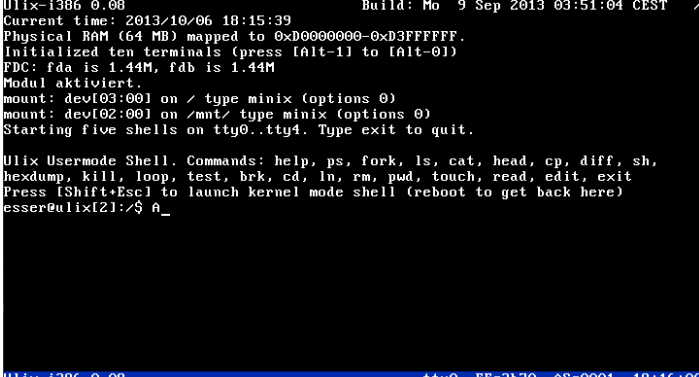
- **Multi-User-Betrieb:**
 - ✓ Datenstrukturen vorhanden
 - ✗ effektiv kein Multi-User-Betrieb möglich, keine Rechteprüfung, keine Anmeldung (immer root) (ab v. 0.12: Login, su)
- **Terminals:**
 - ✓ Ulix startet Shells auf mehreren Terminals
 - ✗ kein Grafikmodus; Terminals laufen in 80x24-Textmodus (Zeile 25: BS-Statuszeile)

ULIX: Features, Einschränkungen (4)

- **Zugriff auf Geräte:**
 - ✓ Floppy: DMA, Block-Transfer
 - ✓ Festplatte: PIO, Block-Transfer
 - ✓ „Serial Harddisk“: byte-weiser Transfer über serielle Schnittstelle
 - ✗ keine Gerätedateien (/dev/*), ab 0.12 vorhanden
- **Netzwerk:**
 - ✗ nicht implementiert (in BA: IP via SLIP)

Ulix 0.08 Statistik

```
$ wc -lc ulix.c printf.c start.asm
 9155 283752 ulix.c      // Kernel
   185   4036 printf.c  // printf-Implement.
   355   9244 start.asm  // booten, IRQs
 9695 297032 total
```



```
Ulix-i386 0.08                               Build: Mo  9 Sep 2013 03:51:04 CEST
Current time: 2013/10/06 18:15:39
Physical RAM (64 MB) mapped to 0x00000000-0x03FFFFFF.
Initialized ten terminals (press [Alt-1] to [Alt-0])
FDC: fda is 1.44M, fdb is 1.44M
Modul aktiviert.
mount: dev[03:00] on / type minix (options 0)
mount: dev[02:00] on /mnt type minix (options 0)
Starting five shells on tty0..tty4. Type exit to quit.

Ulix Usermode Shell. Commands: help, ps, fork, ls, cat, head, cp, diff, sb,
hexdump, kill, loop, test, brk, cd, ln, rm, pud, touch, read, edit, exit
Press [Shift+Esc] to launch kernel mode shell (reboot to get back here)
esser@ulix[2]:/$ _
```