Hacking your own window manager

sECuRE auf der GPN 8

powered by LaTeX, of course

27. Juni 2009

Dieser Vortrag

- Geschichte/Einführung in Window Manager
- Merkmale von i3
- Window Manager und X11
- Arbeitsumgebung
- XCB
- Setup
- Reparenting (Window Decorations)
- Events
- Hints (Titel, Klassen, Größen, . . .)
- Gotchas
- Zusammenfassung

Geschichte/Einführung

- „All window managers suck, this one just sucks less"?
- Desktop environment vs. window manager (GNOME, KDE, Xfce, ...)
- Stacking (e17, fluxbox, IceWM, fvwm, ...) vs Tiling (dwm, wmii, xmonad, ...)
- dwm, awesome, xmonad, ...: statisches Layout
- wmii, ion: dynamisches layout
- Probleme an ion: tuomov (Lizenz, Kommunikation), Config, Look and feel, Code
- Probleme an wmii: Xinerama-support, Xlib, undokumentierter Code, nur Spalten, keine Reihen, Kleinigkeiten (titellose Fenster)

Merkmale von i3

- gut lesbarer, dokumentierter Code. Dokumentation.
- XCB anstelle von Xlib
- Xinerama done right™
- Spalten und Zeilen, Tabelle als Basis
- command-mode, wie in vim
- UTF-8 clean
- kein Antialiasing, schlank und schnell bleiben

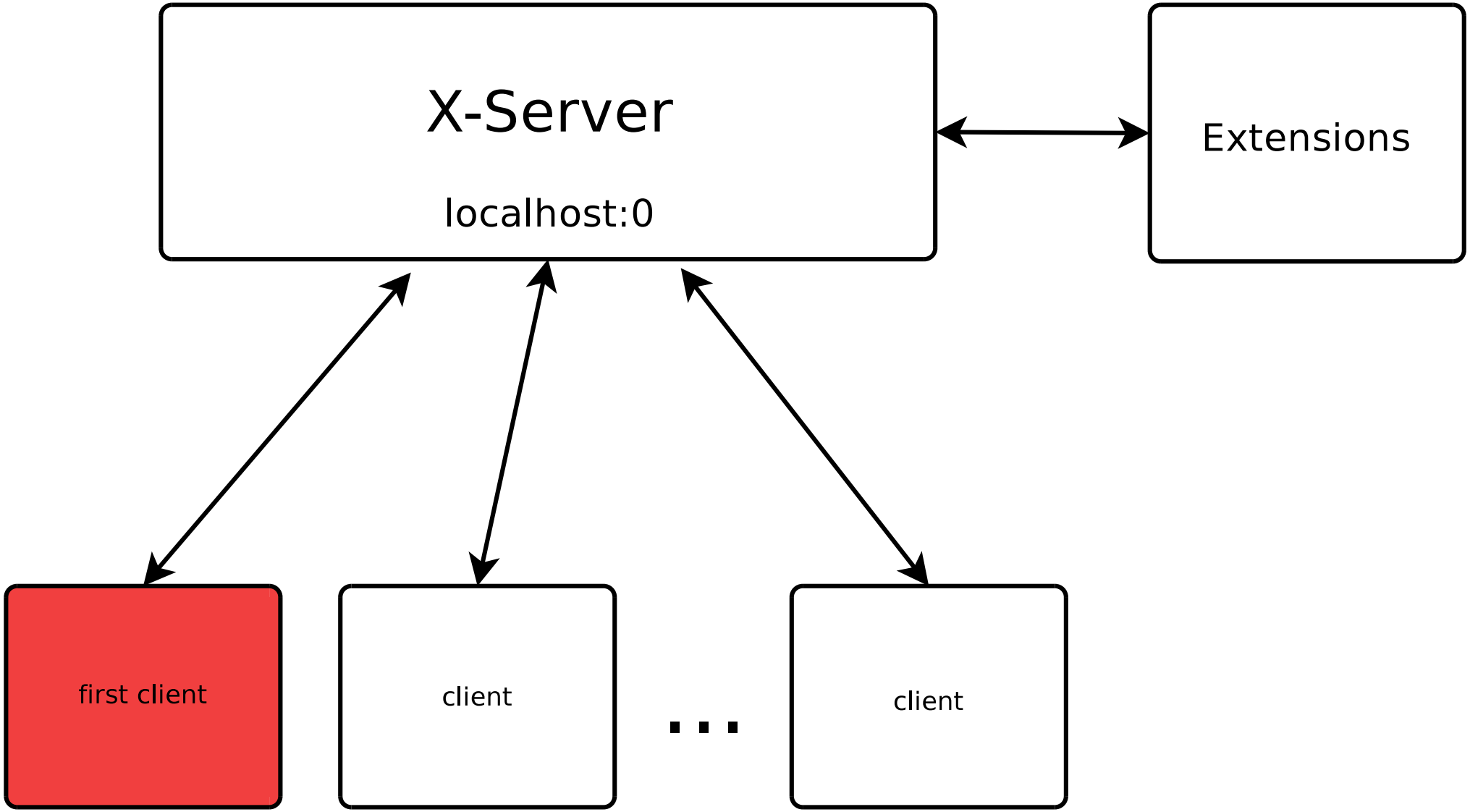
Typische Kommunikation mit X

- Verbindung aufbauen
- Requests über die Leitung schicken (Fenster erzeugen)
  - Cookie für jeden Request
  - Antwort für spezifisches Cookie abholen
  - $\Rightarrow$ Asynchronität nutzbar
- Eventloop starten, reagieren (Fenster zeichnen, Eingaben, . . .)

Was genau macht ein WM?

- Events umlenken
- Neue Fenster anzeigen/positionieren (MapRequest)
- Titelleisten malen (reparenting)
- Den Fokus verwalten
- Mit Hints umgehen (Fenstertitel, Fullscreen, Dock, . . .)
- Auf Benutzereingaben reagieren

Window Manager und X11 (1)



Window Manager und X11 (2)

- Keine Rechteaufteilung, prinzipiell kann jeder Fenster managen
- Window Manager verantwortlich für alle Kinder das Root-Fensters
- RedirectMask, lässt sich Events des Root-Fensters schicken
- Setzt hints auf dem Root-Fenster

Arbeitsumgebung

- X sinnvoll beim Entwickeln ⇒ anderen Computer verwenden oder Xephyr
- xtrace dazwischenschalten (sowohl zwischen WM und X11 als auch zwischen Clients und X11 sinnvoll)
  `DISPLAY=:1 xtrace -o /tmp/xtrace.log -n :9`
- `xprop` zeigt Hints an, `xwininfo` gibt Struktur aus
- als ersten Client ein Terminal starten ⇒ wenn der WM crashed lebt die X-Session noch
  `DISPLAY=:1 urxvt &`
- Debugger, strace, logfiles, core-dumps aktivieren
  (Siehe auch `http://i3.zekjur.net/docs/debugging.html`)

XCB

- `http://xcb.freedesktop.org/`
- „X-protocol C-language Binding”
- Klein, wartbar (aus einer Protokollbeschreibung auto-generiert)
- Sinnvoll benannte Funktionsnamen und Datentypen
- Nutzt die Asynchronität von X aus
- Allerdings: Sehr spärlich dokumentiert, man muss mit Xlib-Doku arbeiten
- xcb-util: XCB noch mal ein bisschen gekapselt, nützliche Funktionen abstrahiert

Xlib-Beispielcode

```
1  char *names[10] = {"_NET_SUPPORTED", "_NET_WM_STATE",
2  "_NET_WM_STATE_FULLSCREEN", "_NET_WM_NAME" /* ... */};
3  Atom atoms[10];
4
5  /* Get atoms */
6  for (int i = 0; i < 10; i++) {
7    atoms[i] = XInternAtom(display, names[i], 0);
8  }
```

XCB-Beispielcode

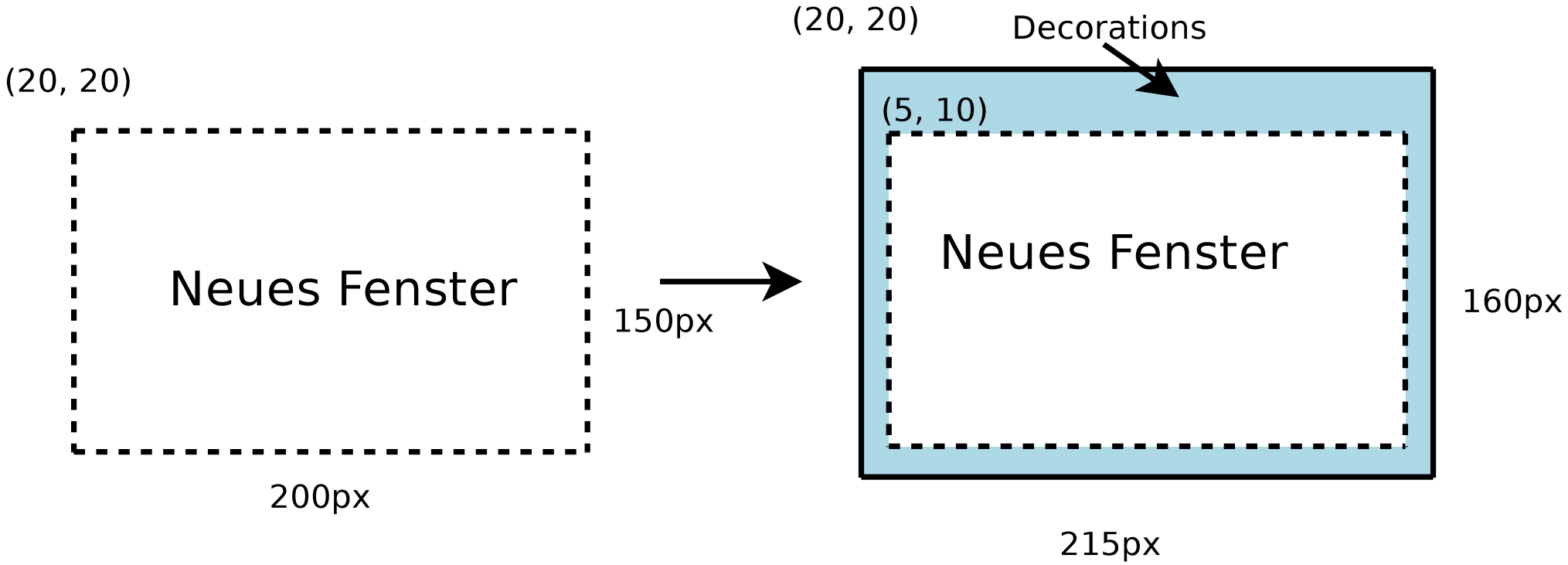
```
1  char *names[10] = {"_NET_SUPPORTED", "_NET_WM_STATE",
2    "_NET_WM_STATE_FULLSCREEN", "_NET_WM_NAME" /* ... */};
3  xcb_intern_atom_cookie_t cookies[10];
4
5  /* Place requests for atoms as soon as possible */
6  for (int c = 0; c < 10; c++)
7    xcb_intern_atom(connection, 0, strlen(names[c]), names[c]);
8
9  /* Do other stuff here */
10 load_configuration();
11
12 /* Get atoms */
13 for (int c = 0; c < 10; c++) {
14   xcb_intern_atom_reply_t *reply =
15     xcb_intern_atom_reply(connection, cookies[c], NULL);
16   if (!reply) {
17     fprintf(stderr, "Could not get atom\n");
18     exit(-1);
19   }
20   printf("atom has ID %d\n", reply->atom);
21   free(reply);
22 }
```

Setup

```
1  get_atoms();
2
3  xcb_event_set_key_press_handler(&evenths, handle_key_press, NULL);
4  xcb_property_set_handler(&prophs, WM_TRANSIENT_FOR, UINT_MAX,
5                           handle_transient_for, NULL);
6
7  xcb_grab_key(conn, 0, root, modifier, keycode,
8               XCB_GRAB_MODE_SYNC, XCB_GRAB_MODE_ASYNC);
9  xcb_grab_key(conn, 0, root, modifier | xcb_numlock_mask, keycode,
10              XCB_GRAB_MODE_SYNC, XCB_GRAB_MODE_ASYNC);
11
12 uint32_t values[] = { XCB_EVENT_MASK_SUBSTRUCTURE_REDIRECT |
13                       XCB_EVENT_MASK_STRUCTURE_NOTIFY |
14                       XCB_EVENT_MASK_PROPERTY_CHANGE |
15                       XCB_EVENT_MASK_ENTER_WINDOW };
16 xcb_change_window_attributes(conn, root, XCB_CW_EVENT_MASK, values);
17
18 manage_existing_windows();
19
20 xcb_event_wait_for_event_loop(&evenths);
```

Siehe auch: i3/src/mainx.c:370ff

Reparenting



1. (App) Fenster wird konfiguriert (Position, Größe, . . .)
2. (App) MapRequest
3. (WM) Window Manager erstellt eigenes Fenster
4. (WM) Reparent = neues Fenster kriegt statt root das WM-Fenster als parent
5. (WM) Mappen des neuen Fensters

fake_configure_notify

- (Alte) Reparented clients kriegen nichts mit, denken relativ zum root-Fenster
- $\Rightarrow$ Window Manager tut so, als würde das Fenster neu konfiguriert, sendet den Event mit absoluten statt relativen Koordinaten
- Sieht man sehr gut an `xfontsel` und anderen Anwendungen, die Xaw (X Athena widget set) verwenden

```
1        xcb_configure_notify_event_t generated_event;
2        generated_event.window = window;
3        generated_event.response_type = XCB_CONFIGURE_NOTIFY;
4        generated_event.x = r.x;
5        /* ... */
6        generated_event.override_redirect = false;
7        xcb_send_event(conn, false, window,
8                       XCB_EVENT_MASK_STRUCTURE_NOTIFY,
9                       (char*)&generated_event);
```

Siehe auch: i3/src/xcb.c:193ff

Events: button_press

- Aktiv grabben, die Anwendung soll keinen Klick bekommen, wenn der Nutzer das Fenster verschiebt

```
1  int handle_button_press(void *ignored, xcb_connection_t *conn,
2                          xcb_button_press_event_t *event) {
3          /* ... */
4          if ((event->state & XCB_MOD_MASK_1) != 0)
5                  floating_drag_window(conn, client, event);
6
7          /* ... */
8          if (event->detail == XCB_BUTTON_INDEX_4 ||
9              event->detail == XCB_BUTTON_INDEX_5) {
10                 LOG("User scrolled\n");
11                 return 1;
12         }
13         /* if unhandled, forward the click to the application */
14         xcb_allow_events(conn, XCB_ALLOW_REPLAY_POINTER, event->time);
15         return 1;
16 }
```

Siehe auch: i3/src/handlers.c:148ff

Events: enter_notify

- Der Mauszeiger ist über dem Fenster gelandet
- Auch unabsichtlich: Wenn das Fenster unter den Mauszeiger konfiguriert wird
- ⇒ Blacklist an Events, die man ignorieren muss

```
1  int handle_enter_notify(void *ignored, xcb_connection_t *conn,
2                          xcb_enter_notify_event_t *event) {
3          if (event_is_ignored(event->sequence))
4                  return 1;
5
6          Client *client = table_get(&by_parent, event->event);
7          if (client == NULL) {
8                  return 1; /* user moved cursor to another screen */
9          }
10
11         set_focus(conn, client, false);
12         return 1;
13 }
```

Siehe auch: i3/src/handlers.c:148ff

Events: key_press

- Aktives key grabbing: WM entscheidet, ob Tastendruck weitergeht, also bei der Anwendung ankommt (kann abfangen)
- Passives key grabbing: WM kriegt einen event

```
1  uint16_t state_filtered =
2           event->state & ~(xcb_numlock_mask | XCB_MOD_MASK_LOCK);
3  state_filtered &= 0xFF; /* filter mouse buttons */
4  foreach (binding) {
5          if (binding->keycode == event->detail &&
6              binding->mods == state_filtered) {
7                  /* do fancy stuff here */
8                  break;
9          }
10 }
```

Siehe auch: i3/src/handlers.c:100ff

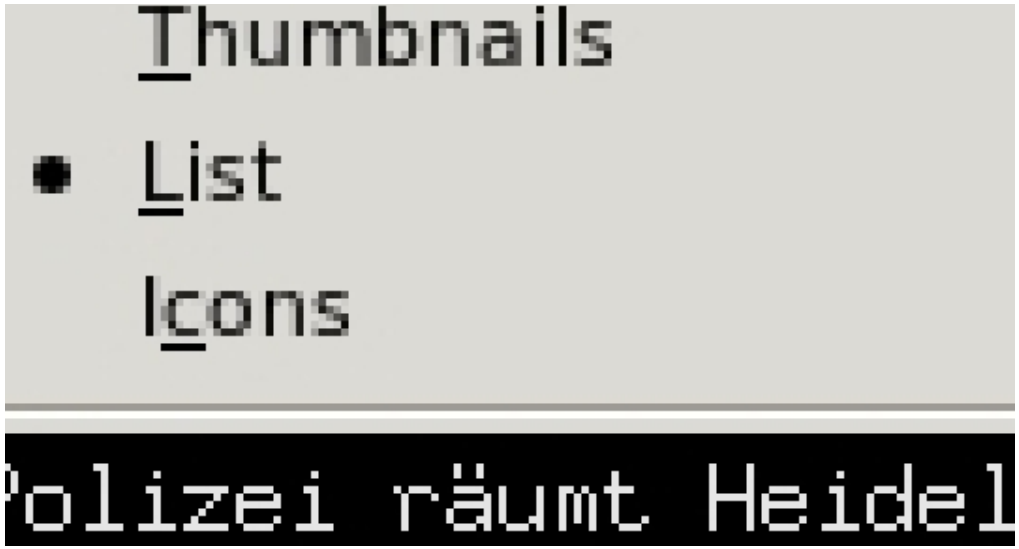
Events: key_press (2), Mode_switch

- `event->state` enthält nie das Mode_switch-Bit, Bug in X
- XKB hilft, den korrekten state zu ermitteln
- ⇒ Mode_switch nicht als modifier in `xcb_grab_key` verwendbar
- ⇒ wir grabben alle keys aktiv (!) und filtern selbst nach Mode_switch

```
1  /* ... state_filtered is already cleaned */
2  XkbStateRec state;
3  if (XkbGetState(xkbdpy, XkbUseCoreKbd, &state) == Success &&
4      (state.group+1) == 2)
5          state_filtered |= BIND_MODE_SWITCH;
6  foreach (binding)
7          if (binding->keycode == event->detail &&
8              binding->mods == state_filtered) {
9                  xcb_allow_events(conn, SyncKeyboard, event->time);
10                 return; /* after doing actual stuff, of course */
11         }
12 xcb_allow_events(conn, ReplayKeyboard, event->time);
```

Siehe auch: i3/src/handlers.c:100ff

Umlaute und Sonderzeichen



- Verschiedene APIs für Render von Text: X Core Fonts und xft
- xft = X FreeType, antialiased fonts, Pango, GTK
- Problem mit X Core Fonts: keine Sonderzeichen
- ...oder? `misc-fixed-*-iso10646`, also X Core Fonts mit Universal Character Set (= Unicode-Zeichen). Nicht 100% vollständig
- urxvt: benutzt beide APIs, pro Glyph unterschiedlich
- Trend geht leider zu fontconfig/xft :-(

Umlaute und Sonderzeichen (2)

- X hat eigenes Encoding: Compound Text
- Früher ICCCM (Compound text, z.B. Atom WM_NAME)
  ICCCM = Inter-Client Communication Conventions Manual
- heute EWMH (UTF-8, z.B. Atom _NET_WM_NAME)
  EWMH = Extended Window Manager Hints (= NetWM)
- XImageText16 (bzw xcb_image_text_16) erwartet UCS-2
  ⇒ `iconv_open(UCS2_BE, UTF-8)`

`Siehe auch: i3/src/util.c:191ff, i3/src/handlers.c:663ff`

Colorpixel

- Heutzutage: TrueColor. Früher: 8 bit o.ä.
- Colormaps: Geben an welche Farben die Hardware kann
- Colorpixel: Ein Wert aus der Colormap, der der gewünschten Farbe am nähesten kommt
- Bei TrueColor: `return (red << 16) + (green << 8) + blue;`
- Alles andere: Round-Trip zum X-Server:

```
1         #define RGB_8_TO_16(i) (65535 * ((i) & 0xFF) / 255)
2         xcb_alloc_color_reply_t *reply;
3         reply = xcb_alloc_color_reply(conn, xcb_alloc_color(conn,
4                 root_screen->default_colormap, RGB_8_TO_16(red),
5                 RGB_8_TO_16(green), RGB_8_TO_16(blue)), NULL);
6         if (!reply)
7                 die("Could not allocate color\n");
8         return reply->pixel;
```

Siehe auch: i3/src/xcb.c:76ff

Hints

- NetWM

  NET_WM_WINDOW_TYPE dock, dialog, utility, toolbar, splashscreen
  NET_WM_NAME Fenstertitel (UTF-8), auch auf dem root-Fenster
  NET_WM_STRUT_PARTIAL Reservierter Bereich am Bildschirmrand (Docks), z.B. für dzen2

- ICCCM

  WM_NAME Fenstertitel (Compound Text)
  WM_TRANSIENT_FOR Zugehöriges, „temporäres“ Fenster für Anwendung X ($\Rightarrow$ floating)
  WM_CLASS Fensterklasse (z.B. „urxvt“), praktisch zum identifizieren
  WM_NORMAL_HINTS (Size hints), beinhaltet Aspect Ratio (mplayer!), minimale und maximale Größe

Hints (2)

```
1  int handle_transient_for(void *data, xcb_connection_t *conn,
2              uint8_t state, xcb_window_t window,
3              xcb_atom_t name, xcb_get_property_reply_t *reply)
4  {
5    xcb_window_t transient_for;
6    if (reply != NULL) {
7      if (!xcb_get_wm_transient_for_from_reply(&transient_for, reply)) {
8        LOG("Not transient for any window\n");
9        return 1;
10     }
11   } else {
12     if (!xcb_get_wm_transient_for_reply(conn,
13         xcb_get_wm_transient_for_unchecked(conn, window),
14         &transient_for, NULL)) {
15       LOG("Not transient for any window\n");
16       return 1;
17     }
18   }
19   if (client->floating == FLOATING_AUTO_OFF)
20     toggle_floating_mode(conn, client, true);
21   return 1;
22 }
```

Gotchas

- Flushing (`xcb_flush(connection);`)
- `WM_STATE` != `WM_STATE_NORMAL`
- Eventloops / Caching von xcb (GIMP splash screen)

Zusammenfassung

- Bindings aufsetzen, Eventmask konfigurieren
- Events/Hints abarbeiten
- Decorations zeichnen

Lust bekommen?

- git clone `git://code.stapelberg.de/i3`
- development branch: `git checkout --track -b next origin/next`
- Debian: `apt-get install i3-wm/unstable`
- non-Debian: `cd i3; cat DEPENDS; make && sudo make install`
- in ~/.xsession: `exec /usr/bin/i3`
- Siehe manpage `i3(1)`, user's guide, how to hack

exit(0);

- git-webinterface: `http://code.stapelberg.de/git/i3`
- Website: `http://i3.zekjur.net`
- IRC: #i3 auf irc.twice-irc.de
- xcb: `http://xcb.freedesktop.org/`
- 50-Zeilen-WM: `http://incise.org/tinywm.html`
- „Why X is not our ideal window system”: `http://www.std.org/~msm/common/WhyX.pdf`
- . . . noch Fragen?