System development

# System development

-----------------------------------------------------------------------------------------------------------------------

Использование инструментов операционной системы для разработки приложений

------------------------------------------------------------------------------------------------------------------------

Такого термина не существует на самом деле, но я его придумал и он довольно точно описывает область проблем, связанных с разработкой приложений с учетом важных особенностей операционной системы (ядро линукс), применяющий возможности и компоненты ОС (контейнеризация итд) и взаимосвязи компонентов. Важно это учитывать при разработки приложений и методов разработки приложений.

Релевантные реализные термины:

| **Область** | **Краткое описание** | **Связь с System Development** | **Используемые технологии / инструменты** |
| --- | --- | --- | --- |
| **Linux Systems Programming** | Программирование с использованием системных вызовов, управление процессами, памятью, IPC | ✅ Базовый фундамент: ядро, namespaces, сокеты, безопасность | C, Rust, ptrace, clone(), mmap, epoll, seccomp |
| **Container Engineering** | Внутреннее устройство и управление контейнерами, runtime, изоляция | ✅ Контейнеризация и работа с окружением | Docker, Podman, LXC, runc, containerd, OCI |
| **Cloud-Native Backend Development** | Разработка API и сервисов, работающих в контейнерах и Kubernetes | ✅ Приложения, которые "знают" про среду и адаптируются | Go, gRPC, REST, Envoy, service mesh, OpenAPI |
| **Interprocess Communication (IPC)** | Взаимодействие процессов через сокеты, пайпы, shared memory, D-Bus | ✅ Ключ для связи между контейнерами и хостом / GUI | Unix Sockets, D-Bus, gRPC, named pipes, ZeroMQ |
| **Remote GUI / Containerized UI** | Запуск и вывод GUI из контейнера в хост, проброс GPU / звука / окна | ✅ Уникальная ниша, сложная, но нужная | X11, Wayland, PulseAudio, VNC, NVIDIA Container Toolkit |
| **Platform / Infrastructure Engineering** | Создание среды, в которой работают контейнеры и backend-приложения | ⚠️ Чуть вышеуровнево, но часто пересекается | Kubernetes, Helm, Terraform, systemd, observability stack |
| **Security & Sandboxing** | Ограничение привилегий и контроль изоляции в Linux | ✅ Важный аспект при контейнеризации и GUI | AppArmor, SELinux, seccomp, capabilities, Flatpak |
| **Embedded Linux / Edge Systems** | Разработка в ограниченных или нестандартных окружениях | ✅ Очень схоже: ресурсы, окружение, вывод UI | Buildroot, Yocto, Qt, DRM/KMS, EGLFS |

Interprocess Communication (IPC)

# Передача GUI в другой контейнер

При разработке контейнерных приложений иногда требуется передача визуальной информации в другой контейнер (процесс).

## X11 Server

сайт: <https://x.org/wiki/>

Документация: <https://x.org/wiki/Documentation/>

X — это архитектурно-независимая система для удаленных графических пользовательских интерфейсов и возможностей устройств ввода. Каждый человек, использующий сетевой терминал, имеет возможность взаимодействовать с дисплеем с помощью любого типа устройства ввода пользователя.

Это означает, что:

X не привязан к конкретному CPU или архитектуре, oн может работать:

на Linux на x86,

на BSD на ARM,

на Solaris на SPARC и т.д.