一、课程定位与总体目标

- **操作系统是计算机系统的枢纽**：充当用户与硬件的中介，既需简化使用体验，又要高效调度资源。
- **课程愿景**：让学生从源码层面理解 OS 原理，培养在大型 C 项目中开发、调试与优化的实战能力。

二、教学安排与评估机制

1. **授课与辅导**
   - 周一讲座、周二在线答疑、首周即开始的两小时 Prac 实验。
2. **考核结构**
   - 三次编程作业：权重依次 15 %/25 %/25 %，主题覆盖进程调度、内核编程、虚拟化等。
   - 期末闭卷两小时，占 35 %。
   - 通过门槛：期末 ≥ 40 %，三次作业总分 ≥ 32.5 %。
3. **工作量提醒**
   - CLI 下 C 编程密集，若指针与 Makefile 基础薄弱需预留补课时间。

三、操作系统核心概念速览

- **三大目标**：执行用户程序、提升易用性、优化硬件利用。
- **双重身份**：资源分配者 + 控制程序；内核是唯一长期常驻部分。
- **系统四层**：硬件 → 操作系统 → 系统/应用程序 → 用户。

四、体系结构与设计范式

1. **内核架构对比**
   - 单体内核：全部服务驻留内核，性能高但耦合重。
   - 分层结构：自下而上抽象，增强可维护性。
   - 微内核：仅保留调度、内存管理、IPC；其他服务移到用户态，可靠性高但通信成本大。
   - 模块化内核：Linux/Solaris 通过可热插拔模块折中。
2. **机制 vs. 策略**
   - 机制定义"怎么做"，策略决定"做什么"，二者分离便于扩展。

五、关键管理子系统

管理面	主要职责	课件出处
进程/线程	创建、撤销、调度、同步、通信	
内存	跟踪分配、换入换出、地址空间隔离	
存储	文件目录、权限、设备映射、备份	
I/O	统一驱动、缓冲、缓存、假脱机	
保护与安全	访问控制、审计、防御 DoS	

（表头仅用竖线分隔，仍属纯文本）

六、系统调用与用户接口

- 系统调用是用户态进入内核的“唯一合法入口”，支持进程控制、文件/设备操作、通信、信息维护等五大类。
- 参数传递常用“寄存器 + 参数块地址”组合，以突破寄存器数量限制。
- CLI/Shell 与 GUI 皆通过系统程序封装底层调用，触控设备进一步扩展多点手势交互。

七、运行时支持：中断、双模式与定时器

- **中断/异常**：硬件事件通过向量表跳转到内核服务例程，保存现场后处理。
- **用户态 ↔ 内核态**：模式位硬件隔离；系统调用/异常进入内核，返回前复位。
- **定时器中断**：限制进程独占 CPU，保障分时系统响应 < 1 s。

八、实战环境与学习支持

- 统一在 **OpenBSD 虚拟机** 上完成开发；第二周分配私有仓库，自动进行风格与相似度检测。
- Prac 与 Online Quiz 不计分，但与作业紧密衔接，**强烈建议按周完成。**
- 学术诚信：严禁代码抄袭与共享，违规将被上报。

九、学习建议

1. **先补 C 语言弱项**：指针、结构体、位运算、Makefile。
2. **搭建编译与调试流程**：熟悉 gcc、gdb、make，尽早在 VM 中跑通 Hello Kernel。
3. **每周节奏**：讲座 → Prac → 自测 Quiz → 阅读课件 → 实施作业，滚动迭代。
4. **多用论坛与答疑时段**：及时暴露与解决 bug，避免临近截止“爆雷”。
5. **注意代码风格与提交历史**：自动脚本会检测；保持小步提交、清晰 commit message。