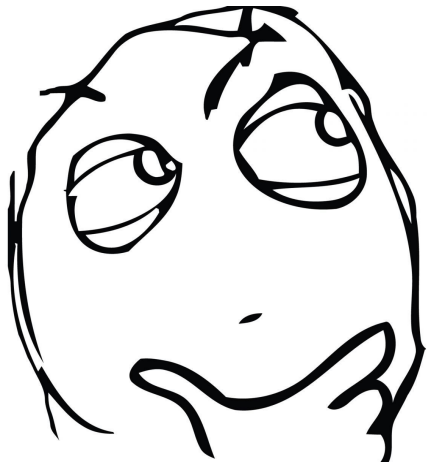
¿Qué hacer antes de las charlas?

"Un par de recomendaciones para tener en cuenta antes de las charlas… y no sentirte bombardeado durante ellas!"



Cátedra de Sistemas Operativos
http://faq.utn.so/como-arrancar
@sisoputnfrba

1) Tener grupo:

- El grupo es de 5 personas
- Busquen gente comprometida, con ganas de ponerle mucha pila.
- ¿Hace falta que sean expertos programadores? **NO**. Todo se aprende, sólo hay que comprometerse.
- Los integrantes pueden ser de distintos cursos? **SI**, *PERO* tienen que tener un turno libre en común los sábados, porque las evaluaciones serán los sábados y requieren que esté el grupo completo

2) Anotarme al Campus Virtual - www.utn.so/campus-virtual

- ¿Cuando? Inmediatamente después de la primera clase
- ¿Por qué? Porque ahí van a estar todas las notificaciones (**cuando se publica el TP**, como anotarme a las entregas, aclaraciones del enunciado, etc).
- "No me convenciste". OK; ahí se pasan las notas. Sin estar registrado en el Campus Virtual no te van a poder firmar la libreta. **Asegurate de tener el ID de curso correcto.**
- MUY IMPORTANTE: suscribite a todos los foros para que te lleguen todos los mensajes por mail.

3) Crear una cuenta en GitHub (https://github.com/join)

- “Ya tengo una cuenta, sirve?” SI; No es necesario crearse una cuenta específica para la cursada, podés usar tu cuenta personal. (Importante: si esa cuenta la usas para trabajar, trata de evitarla)
- Es importante que la cuenta esté confirmada por mail, porque lo requiere el sistema de administración de grupos, y sin eso **no van a poder registrarse en uno.**
- Otra cosa importante (aunque suene tonto aclararlo) es que uses una cuenta de mail activa y a la cual tengas acceso. Si no lo haces, podés llegar a tener problemas con tus commits en github y los ayudantes no vamos a poder tener una comunicación fluida con vos y tu equipo!

4) Usar el foro en Github Issues

- Ahí podés hacer cualquier consulta técnica o sobre el enunciado. Todos los ayudantes nos conectamos de tanto en tanto para responder preguntas referidas al TP.
- Si tu consulta es sobre el diseño de tu solución para el TP, obviamente siempre es obligatorio consultarle a tu ayudante. En otros casos no lo es, pero si lo hacés, es posible que él te pida que eleves la pregunta al foro porque pueda ser útil para otros. En sentido inverso: siempre es bueno revisar lo que se preguntó con anterioridad, puede significarte un ahorro de tiempo!

5) Anotarme en el Sistema de Inscripciones:

- El grupo lo podes ir creando en https://inscripciones.utn.so. Ahi te vas a anotar con tu cuenta de GitHub. Vas a necesitar tu código de curso, así que no hagas esto hasta que tengas confirmada la inscripción a la materia. Acá tenes un video para ver como hacerlo
- “Me puedo anotar con cualquier número de curso si no lo conozco?” Como te dijimos antes, sin estar registrado y, para este caso, en el curso correcto, no te van a poder firmar la libreta. **Asegurate de tener el código de curso correcto.** Tené en cuenta que usamos el sistema para la creación de actas.

6) Leer las Normas del Trabajo Práctico

7) Tener Linux. Opciones: VM de la cátedra (LUbuntu), instalarme en mi PC o bajarme un live CD. ¿Cómo elijo?

- Si quiero tener todo instalado de una **(MUY recomendado)**:
  - Uso la máquina virtual (http://www.utn.so/recursos/maquinas-virtuales/) (Ver en Drive)
- Si tengo miedo de romper todo:
  - Uso la máquina virtual;
  - Wubi

        - LiveCD: pruebo las cosas en un ambiente “descartable”, y ante cualquier problema reinicio y empiezo de cero
    - ¿Quiero que vuele (de rápido, y quizás también de explosión)?
        - Instalalo nativo en la PC

NOTA: *la VM Server es el entorno oficial en que se van a probar las entregas. Te lo vamos a repetir ~infinitas veces: desarrollá como/en donde quieras, pero asegurate siempre de que* ***tu trabajo funcione bien en la VM Server, y de saber manejarla.*** *(Te recomendamos que le pegues una mirada a esto para entender algunos de los motivos)*

8) Familiarizarme con Linux. La GUI se maneja “fácil”, pero acá vamos un paso adelante: amigate con la terminal.

- http://mariobash.utn.so/ (Super Mario Bash)
- http://www.ee.surrey.ac.uk/Teaching/Unix/ (Guía)
- http://web.mit.edu/mprat/Public/web/Terminus/Web/main.html (Juego)
- http://www.overthewire.org/wargames/bandit (Juego)

9) Familiarizarse con el desarrollo

- Aprender el lenguaje C
    - Kernighan & Ritchie, El Lenguaje de programación C
    - http://www.learn-c.org/
- Amigarse con los punteros
    - Manejo de Punteros y Memoria Dinámica en C
- Usar Eclipse (debug)
    - http://faq.utn.so/eclipse-guia-grafica
    - http://www.youtube.com/watch?v=XsefDXRfA9k
- Entender cómo se compila
    - http://www.youtube.com/watch?v=DIY8O0vayUo
    - http://www.youtube.com/watch?v=C93u4HpZ5qI
- Ejercicios
    - Incrementales
    - https://github.com/sisoputnfrba/so-exercises
- Sockets
    - http://beej.us/guide/bgnet/ (en ingles)
    - https://radiosyculturalibre.com.ar/biblioteca/PROGRAMACION/Beej-Programacion-en-Redes.pdf (en español)
- Ejemplos
    - https://github.com/sisoputnfrba/so-commons-library (biblioteca que la cátedra provee, y se pueden sacar varios ejemplos de como hacer TADs y Tests Unitarios, además de empezar a probarlas)
- Entender el manejo de versiones (GIT)
    - http://faq.utn.so/git
    - http://git.io/aprender
    - http://git-scm.com/documentation
    - http://www.youtube.com/watch?v=8O4LFYPY-Ww

- https://www.codeschool.com/courses/try-git

10) No te olvides de Paradigmas, Diseño y/o todas las materias que hayas cursado. Los conocimientos en la carrera son transversales, asi que por más que no es un requerimiento haber cursado estas materias, probablemente puedas aprovechar muchos conceptos y tratar de aplicarlos.

- El TP es largo, y *requiere de un diseño previo.* Un buen diseño y una eficaz delegación de tareas es la clave para no llegar a fin de cuatrimestre reescribiendo módulos enteros. **(*OJO!*** No retrases lo inevitable: En algún momento hay que ensuciarse las manos y ponerse a codear... no le tengas miedo!**)**

- Conceptos de desarrollo como el *polimorfismo,* la *reutilización de código,* la *modularización* y la *generación de abstracciones* no son únicamente divagues intelectuales de hippies post recital de los Beatles y magos oscuros del siglo XV, sino que realmente ayudan a tener un código más seguro, entendible y sin redundancia. Aprovechalos, porque vas a tener que trabajar con lo que desarrolles durante todo el cuatrimestre.

- “Puedo usar tests?” *Gracias hijo, no sabes lo que había esperado escucharte decir eso.* Si! Podés, aunque no es obligatorio, pero te va a ayudar mucho cuando sea necesario debuggear.