Sistemas Operativos

Programação em UNIX - Introdução (teórica)

Rui Maranhão (rma@fe.up.pt)

Agenda

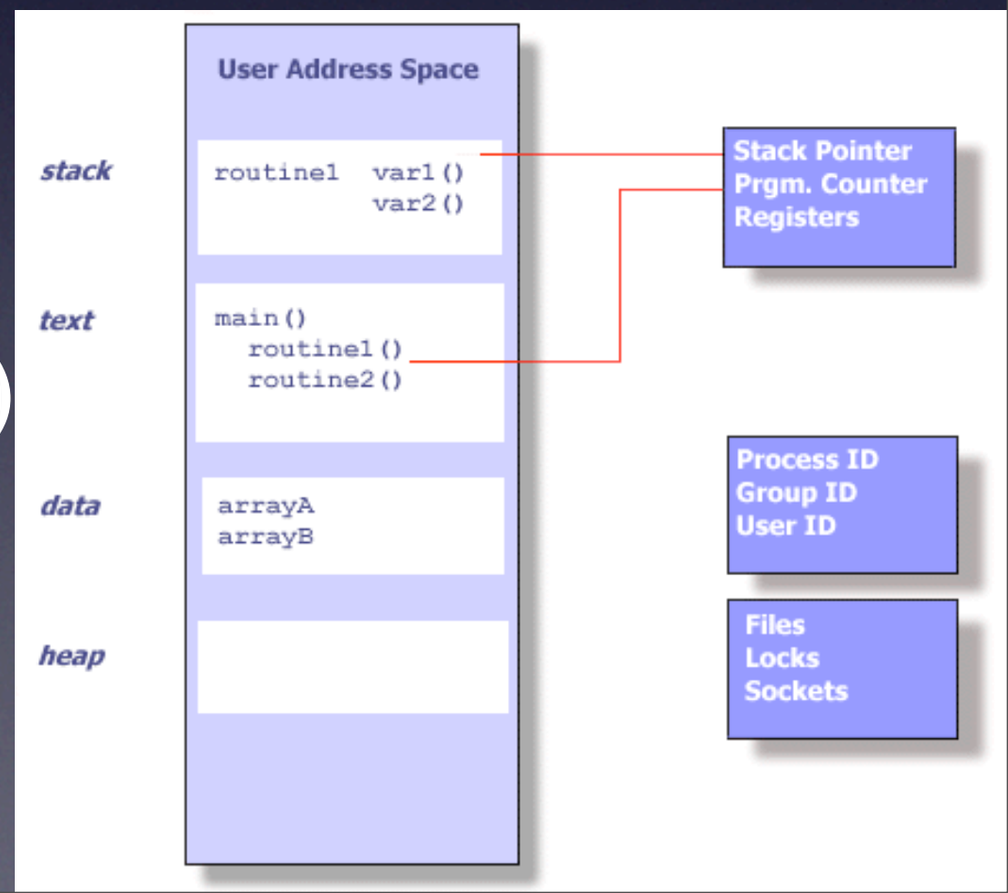
- programa de computador
- introdução aos processos e threads
- do código fonte ao binário
- ferramentas auxiliares
- aplicação make

Programa

- conjunto de instruções e informação para atingir um objectivo (e.g., calcular x^2)
  - instruções e dados
    - (in)dependente de sistema operativo e/ou processador
- quando em execução, designa-se ***processo***

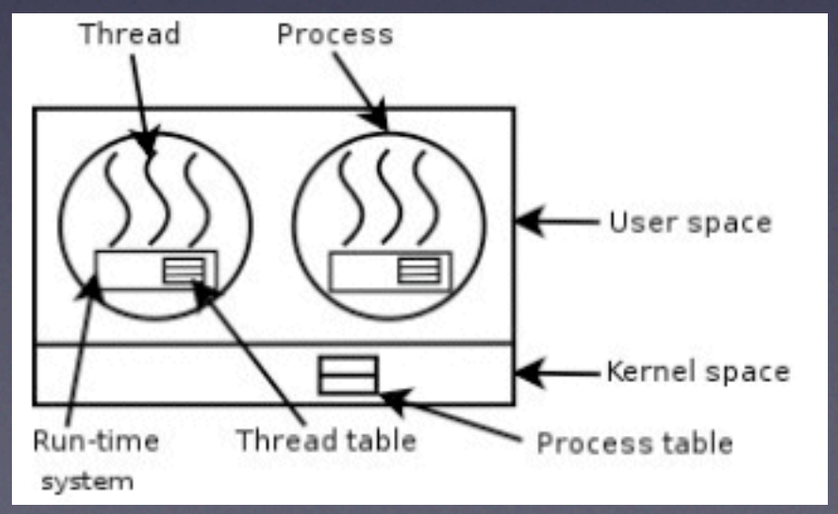
Processos

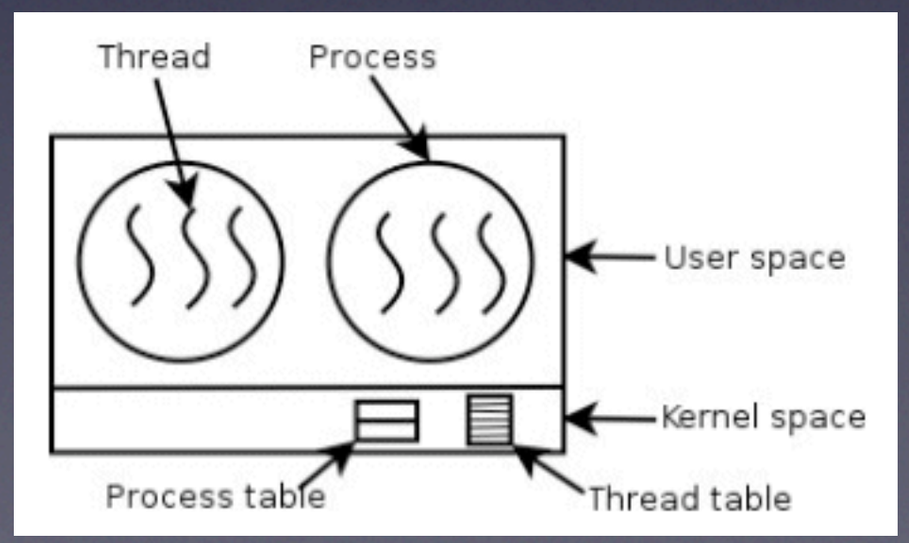
- instância de um programa de computador
- controlado pelo sistema operativo
- representação:
  - espaço de endereçamento
  - variáveis
  - recursos necessários (mutex)



Threads

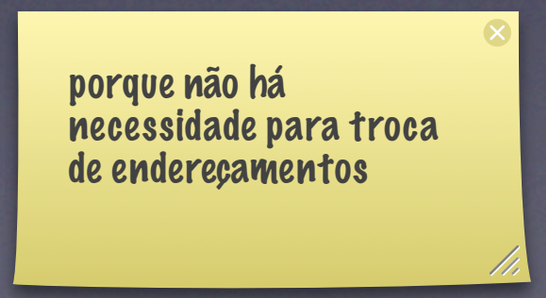
- forma de um processo se dividir em várias tarefas - podem ser executadas em paralelo
- Controlo de threads
- kernel-level thread (fornecido pelo SO)
- user-level thread (fornecido pela LP)





Processos vs. Threads

- criar um processo é caro em termos de tempo/memória/sincronização
- as threads podem ser criadas sem ter de clonar o processo inteiro
- threads são criadas no espaço do usuário
- o tempo gasto para troca de threads é menor



Código Fonte

- ``conjunto de palavras ou símbolos escritos de forma estruturada contendo instruções em uma linguagem de programação”
- C, Java, Perl, Python, .NET, ...
- desenvolvido por programadores (textual)
- executado pelo sistema operativo (binário)

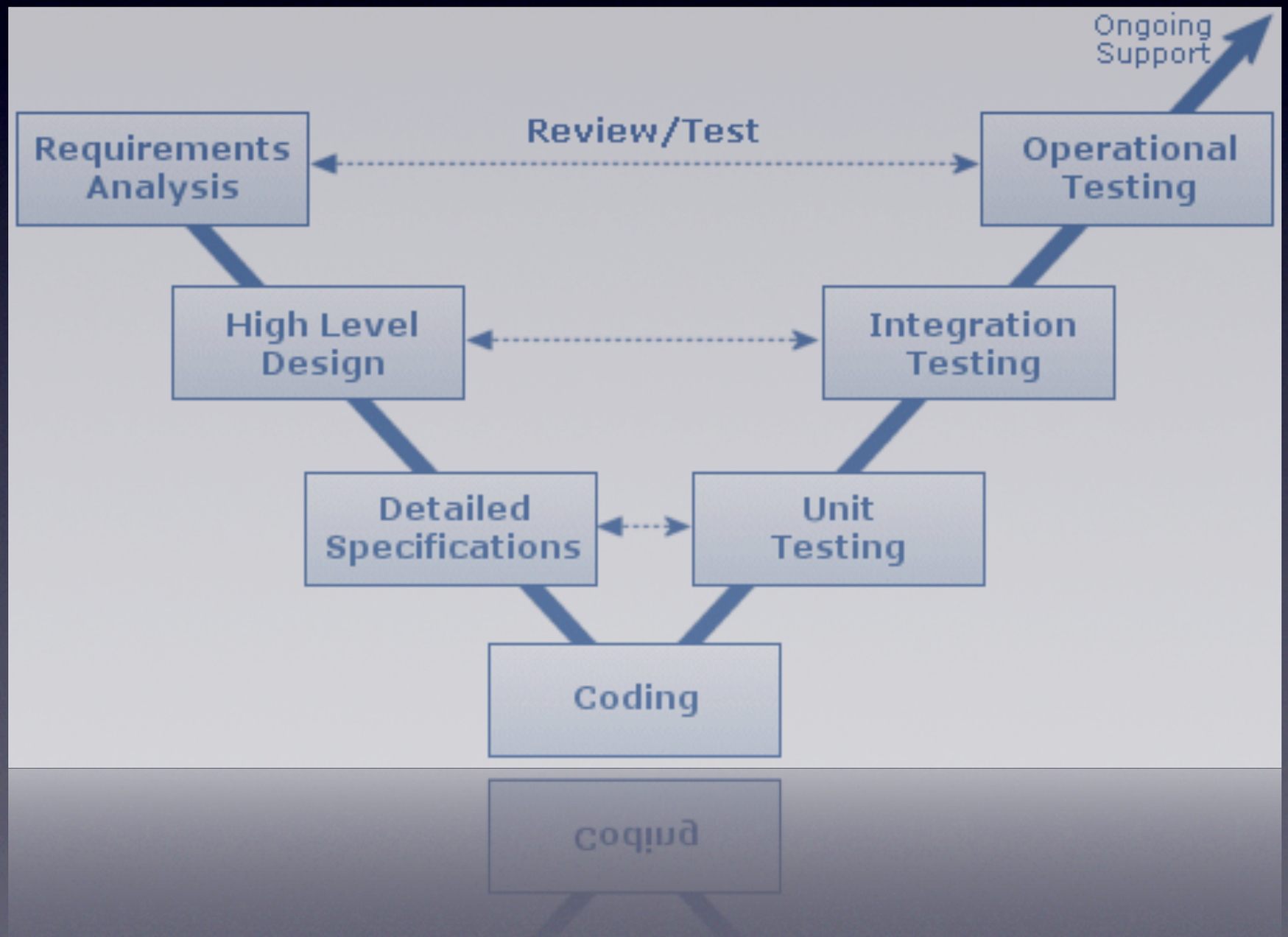
Código Fonte - Estrutura

- partes específicas
- partes gerais
  - declarações (texto)
  - bibliotecas (binários)
    - estáticas
    - dinâmicas ou partilhadas

Bibliotecas dinâmicas

- Prós:
  - partilha espaço de memória
  - actualização automática de aplicações
  - ficheiro executável tem tamanho mínimo
- Cons.:
  - mobilidade
  - desempenho

Conceber Programa



Conceber Programa

- escrever programa - código fonte
  - e.g., kate, xcode, MS Dev Studio
- compilar - código objecto
  - e.g., gcc
- gerar executável - código objecto de bibliotecas
  - linker (e.g., ld)
- executar - código fonte, intermédio ou binário
  - loader, interpretador
- corrigir e aperfeiçoar - código fonte
  - gdb, gcov, gprof, time, ...

Executar Programa

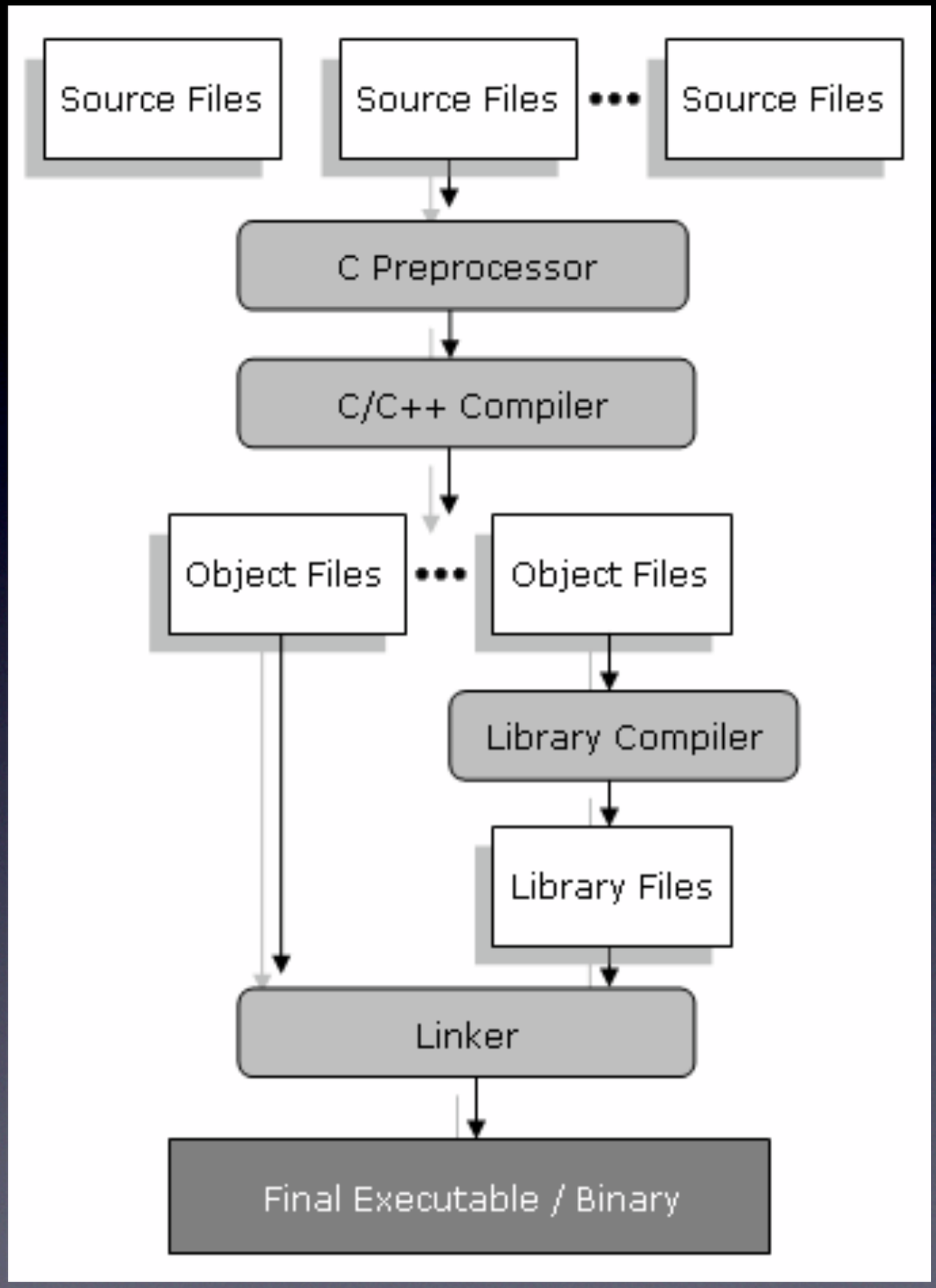
- requisitos necessários
  - código fonte (fontes interpretadas, Bourne shell)
  - código máquina (fontes compiladas, C)
  - código intermédio (fontes semi-compiladas, Java)
- interpretador de comandos
  - SO, loader, linker, hardware, máquina virtual, ...

Debugging

- se o programa falhar, o programador *tem* de encontrar o *bug*
- métodos
  - Manuais: gdb, ddd
  - Automáticos: zoltar (www.fdir.org/zoltar)
- esta é a fase mais cara de todo o ciclo de desenvolvimento de software!!

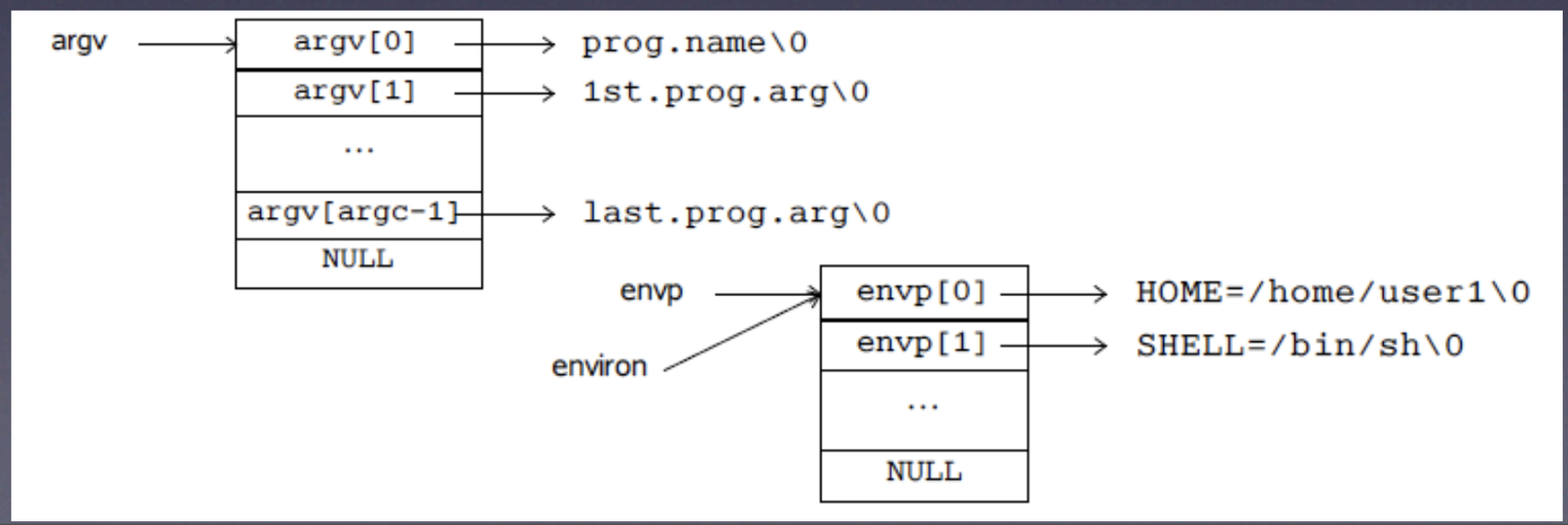
Linguagem de Programação C

- linguagem imperativa
- tipo de dados primitivos: inteiros e reais
- tipo de dados compostos: struct, union,...
- controlo de fluxo: if, for, switch, while,...
- suporte para apontadores
  - char c1, c2, *p;
  - c1 = 'x';
  - p=&c1;
  - c2 = *p

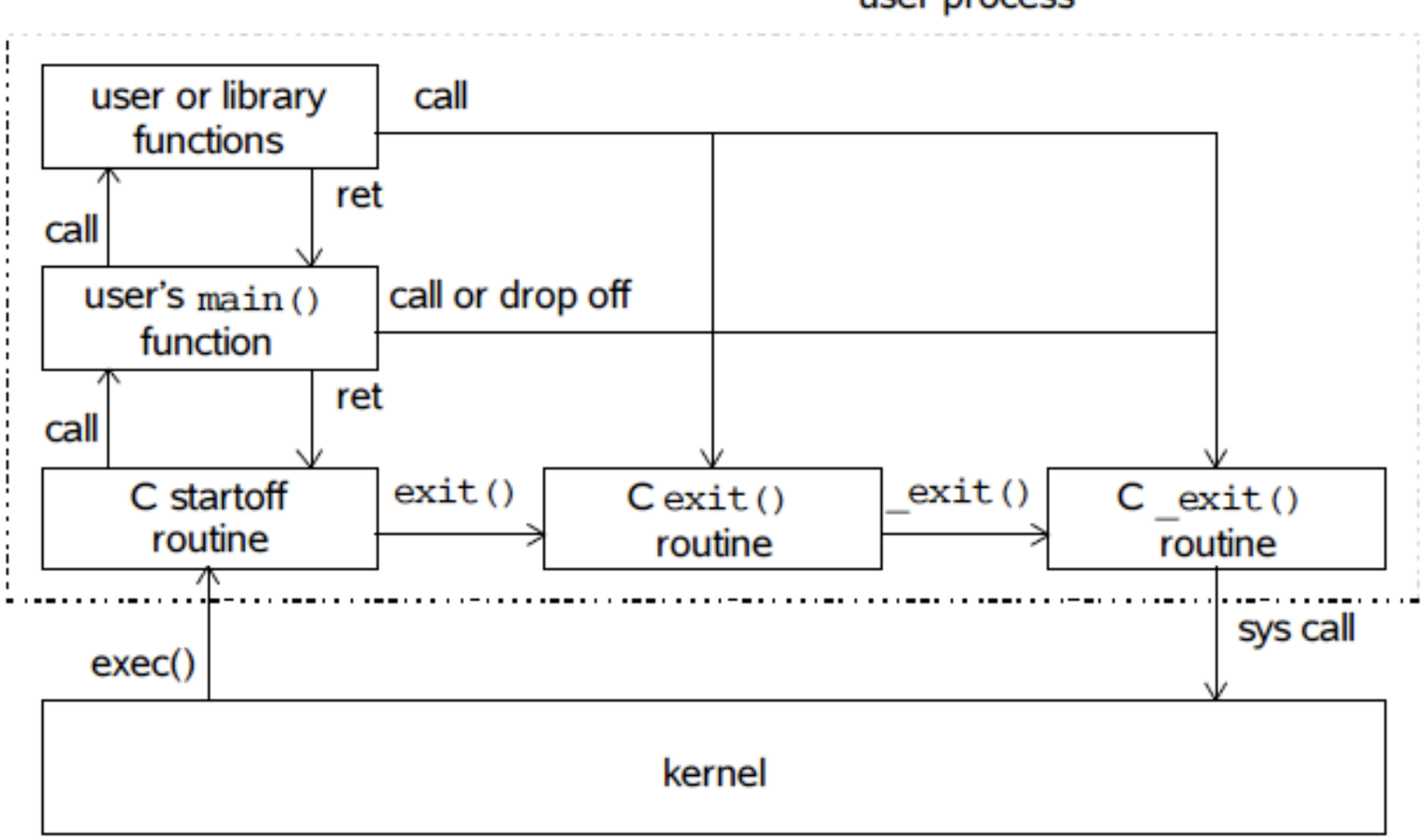

Source Files
Source Files
Source Files
C Preprocessor
C/C++ Compiler
Object Files
Object Files
Library Compiler
Library Files
Linker
Final Executable / Binary

Life-cycle de um programa em C

- arranque
  - int main(int argc, char *argc[], char *envp[])
- terminação
  - void exit(int status)



Life-cycle de um programa em C



Arranque e terminação de um programa em C.

Ferramentas Auxiliares

- ambientes de desenvolvimento (IDE)
  - eclipse, kdevelop
- documentação (ex., man time)
- construção programada (ex., make)
- ambientes de compilação
  - pre-processdor (ex., gcc -E)
  - assemblador (ex., gcc -S)

make

- programa de computador para facilitar compilação de código fonte
- interpretador de programas
  - ficheiro de texto (Makefile)
  - linguagem própria (blocos)
    - produto final, dependências, instruções

make

- Estrutura básica

```
target: dependencies
command 1
...
...
command n
```

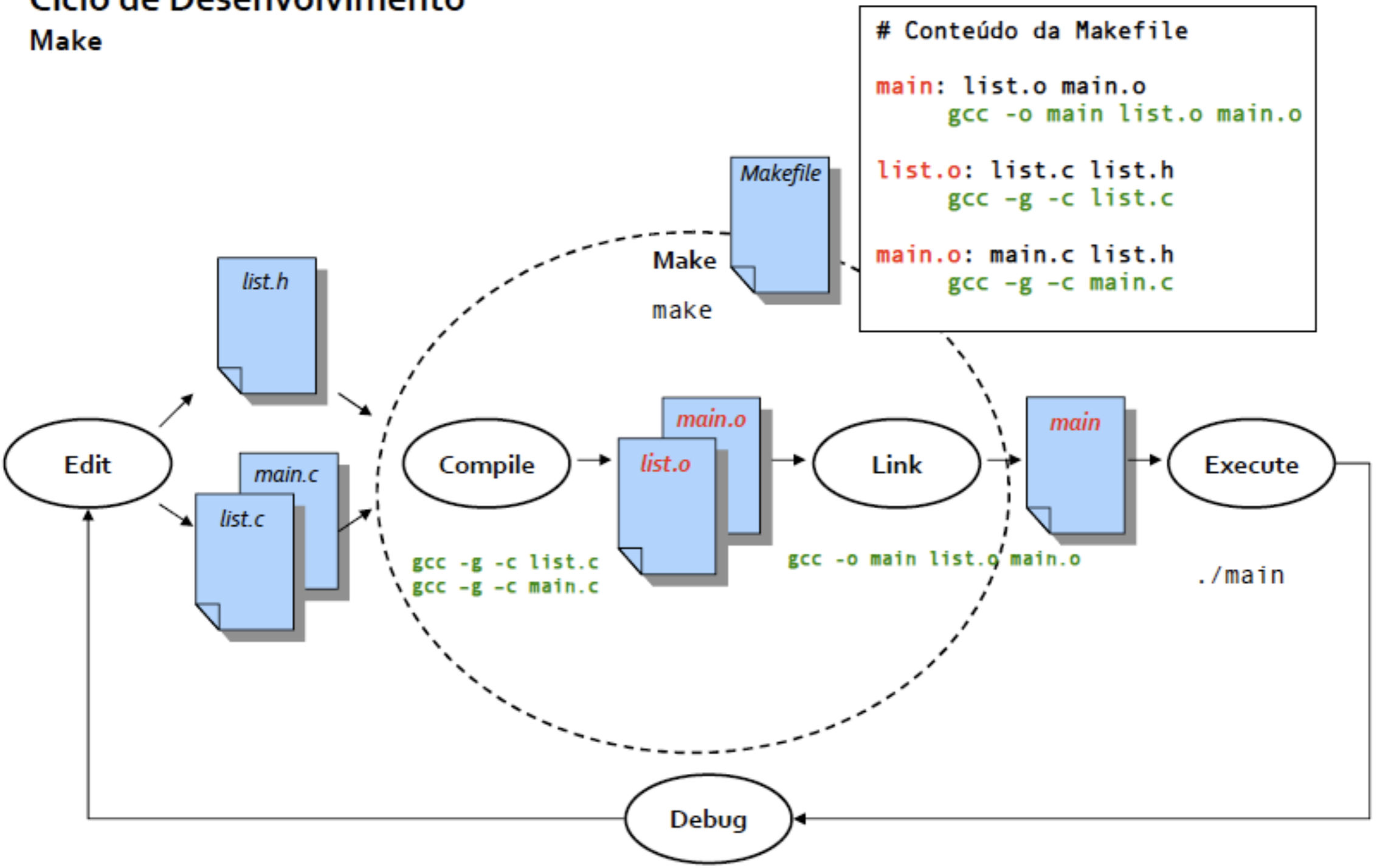
- shell> make

make: exemplo

```
all: hello

hello: main.o factorial.o hello.o
	g++ main.o factorial.o hello.o -o hello

main.o: main.cpp
	g++ -c main.cpp

factorial.o: factorial.cpp
	g++ -c factorial.cpp

hello.o: hello.cpp
	g++ -c hello.cpp

clean:
	rm -rf *o hello
```

Ciclo de Desenvolvimento

Make



sneak-preview TP1

- familiarização com a API do UNIX
- resolução de questões teóricas
- implementação de pequenos programas
- uso de make e gdb

Dúvidas??

- Grupos formados?
  - não é necessário pré-registo de grupos
    - sugestão: alunos do mesmo grupo devem estar inscritos na mesma TP
- TP começam na próxima semana!

- website ainda não disponível...
  - ainda não tenho acesso ao sifeup...