A biblioteca poo\_consola oferece algumas funcionalidades para gestão da consola, nomeadamente, o posicionamento de caracteres numa dada linha e coluna, mudança de cores, etc.

A biblioteca ncurses (cross platform) é a base da biblioteca poo\_consola, pelo que é necessário a sua instalação no ambiente de desenvolvimento e configuração do IDE para o seu uso. As indicações são dadas no restante deste documento. São abordados os temas: instalação e configuração no CLion, Inclusão no projeto, funcionalidades incluídas.

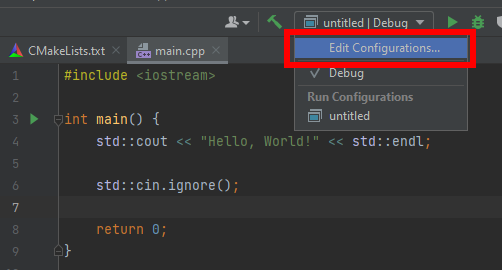
# Configuração do IDE CLion

A biblioteca fornecida para o trabalho usa internamente a biblioteca ncurses que atua sobre o terminal (aquela janela escura onde se escrevem os comandos e que também existe em Windows). Dado que o CLion (e outros IDE) lançam a execução dos programas numa consola própria, torna-se necessário configurar o IDE de forma a lançar a execução dos programas diretamente no terminal no sistema (ou seja “fora do IDE”).

A configuração passa por indicar ao IDE qual é o comando (executável) do sistema que corresponde ao terminal. Este comando varia conforme o sistema, em particular no linux, onde há inúmeras alternativas, conforme a instalação preferida de cada um:

* cmd.exe (windows)
* xfce4-terminal Linux/xfce
* gnome-terminal Linux/gnome
* xterm Linux/X básico
* Terminal Mac

Para executar o programa numa consola fora do IDE, no CLion é necessário alterar as configurações do projeto. A imagem seguinte mostra como se acede a essa configuração no IDE:



A configuração seguinte envolve os campos:

* **Executable**: indica-se o executável que corresponde ao terminal.
* **Program Arguments**: são os argumentos (argumentos da linha de comandos) que serão passados ao programa indicado no campo anterior. Interessa utilizar como argumentos a indicação que o terminal deve iniciar automaticamente a execução do programa correspondente ao projeto.
* **Working directory**: indica qual a diretoria de trabalho do projeto. Interessa indicar aquela onde está o executável do projecto e onde, eventualmente, poderão estar os ficheiros auxiliares envolvidos no trabalho prático.

As informações a colocar nestes campos dependem do sistema operativo.

## Linux

### Target

Desktop XFCE: -> **/usr/bin/xfce4-terminal**

Desktop Gnome: -> **/usr/bin/gnome-terminal**

**Outro desktop**: indicar o binário para o terminal em uso no sistema

**Program Arguments** -> (depende do terminal)

**Desktop Gnome -> gnome-terminal**

**--maximize -- bash -c "./myproj; echo press enter; read" (atenção aos pormenores: “-- bash” -> dois “-” *espaço* “bash”**

**Desktop XFCE /| xfce4/terminal**

**–fullscreen -x bash -c "./myproj; echo press enter; read"**

**substituir myproj** pelo nome (executável) do projeto

### Importante:

* Atenção aos pormenores, tais como o uso de aspas, espaços entre as palavras e símbolos, o uso de ponto-e-vírgulas, etc.

**Ecrã maximizado**: pode-se ser pretendido ter o terminal maximizado (área do ecrã). Se se pretender essa maximização, deve ser feita antes do primeiro uso de ncurses por parte do programa, sendo o melhor o terminal começar já maximizado no lançamento da execução do programa. nas linhas 2 Program Argument indicadas atrás, foram já dadas indicações para tal (que podem ser removidas, caso não se queira ter o terminal maximizado)

* Desktop Gnome / gnome-terminal -> **--maximize**
* Desktop XFCE / xfce4-terminal -> **--fullscreen**

Outros ambientes:

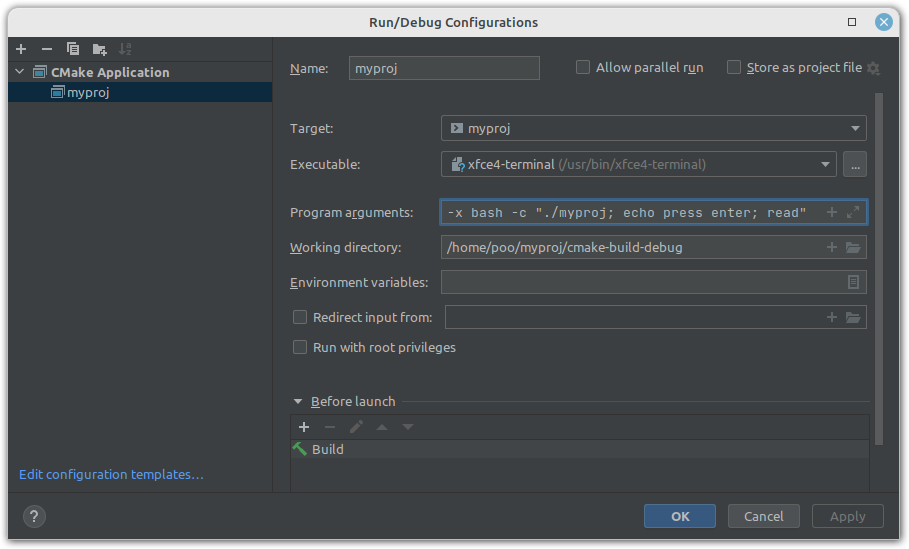
* Windows: opção /MAX
* Mac: não existe forma sistematizável de fazer isto (a gestão de janelas em Mac funciona de forma diferente)

### Working directory -> /home/poo/myproj/cmake-build-debug

**substituir /home/poo/myproj/** pela diretoria base do projeto

**Nota**: a diretoria /home/poo é apenas um exemplo não sendo a habitual para projetos CLion.

Confirmar a configuração com a imagem seguinte (figura no contexto XFCE, sem --fullscreen):



## Windows

Target: -> **c:\windows\system32\cmd.exe**

**substituir c:** pela unidade lógica onde o sistema realmente está instalado

Program Arguments -> **/c start cmd /c "myproj.exe & pause"**

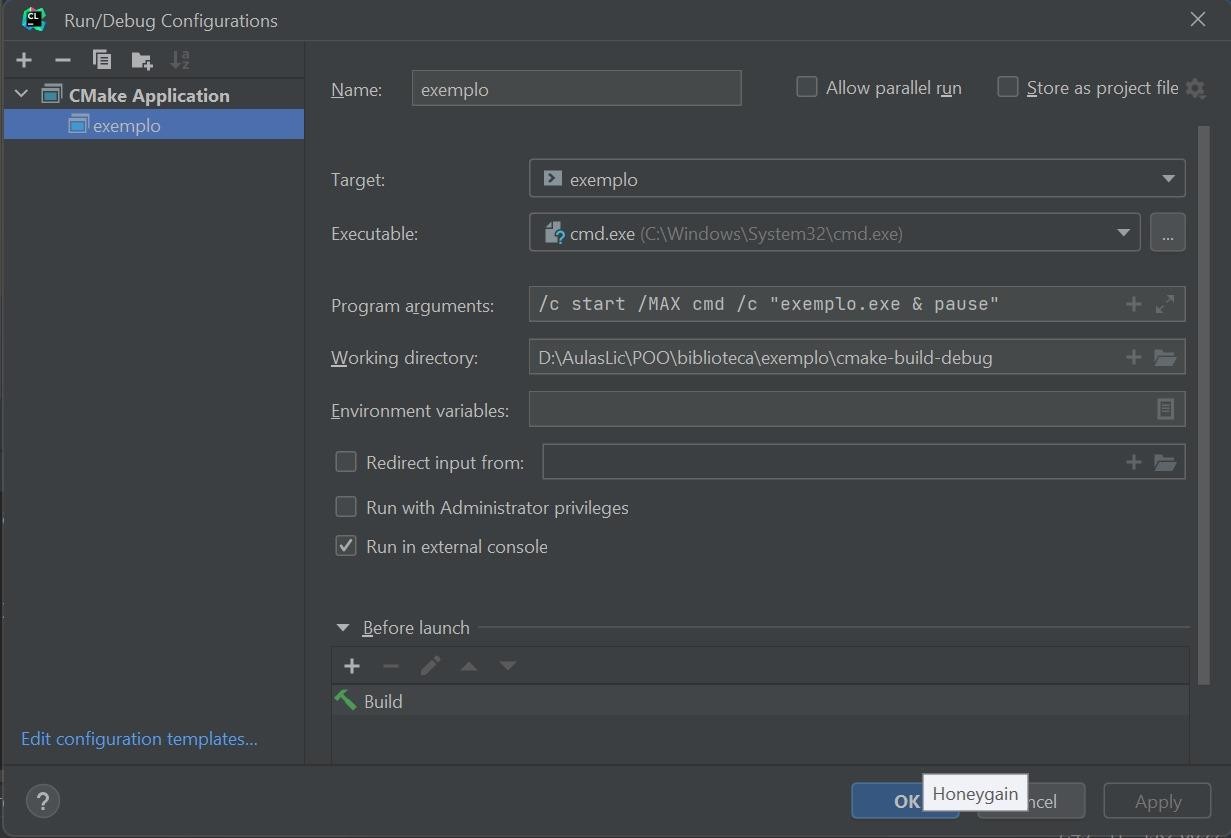
**substituir myproj.exe** pelo nome (.exe) do projeto

Working directory -> **D:\AulasLicPOO\biblioteca\exemplo\cmake-build-debug**

**substituir D:\AulasLicPOO\biblioteca\exemplo** pela diretoria base do projeto Terminal maximizado> Opçãp **/MAX**

**Importante**: após esta configuração o projeto corre num terminal independente do CLIin que é lançado automaticamente por este e que desapareceria automaticamente quando o programa termina. Para se conseguir ver o último output emitido pelo programa é importante que a linha indicada em *program arguments* seja escrita tal como indicado, incluindo os dois /c (não é gralha), as aspas, o caracter & e a menção ao comando pause.

Confirmar a configuração com a imagem seguinte:



## Mac OS

### Target -> /usr/bin/oascript

**Nota**: O terminal (Terminal.app) é invocado através de **osascript**

### Program Arguments -> -e "tell application \"Terminal\" to do script

**\"/home/poo/myproj/cmake-build-debug/myproj; echo press enter; read\""**

**Importante**: trata-se de uma única linha

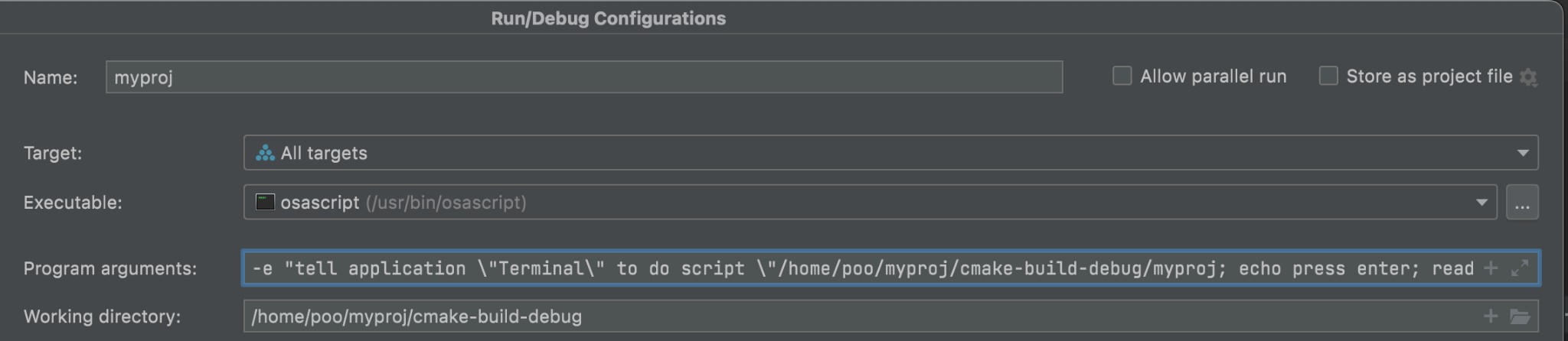
**Substituir /home/poo/myproj/…/myproj** pelo caminho do projeto e o seu respetivo executável

### Working directory -> /home/poo/myproj/cmake-build-debug

**substituir /home/poo/myproj/** pela diretoria base do projeto

**Nota**: a diretoria /home/poo é apenas um exemplo não sendo a habitual para projetos CLion.

Confirmar a configuração com a imagem seguinte:



# Preparação e inclusão da biblioteca no projeto

Dado que a biblioteca fornecida para o trabalho de POO usa internamente a biblioteca ncurses, os binários desta devem ser instalados, de forma a que possam ser usados durante a construção do projeto.

A obtenção e instalação dos binários da biblioteca ncurses depende do sistema, sendo mais fácil em Linux. Deve ser consultada a documentação específica ao sistema operativo/distribuição em uso para pormenores adicionais ou que difiram dos indicados aqui.

Pressuposto

* IDE CLion configurado

O procedimento conforme o sistema onde está a ser desenvolvido o projeto.

**Linux** (debian ou baseado em debian)

1. Instalar as bibliotecas ncurses

Abrir um terminal e executar o comando

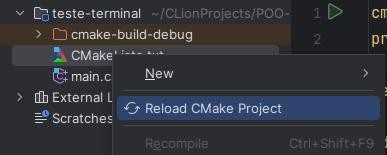
sudo apt-get install libncurses-dev libncursesw5-dev

1. No projeto CLion

Incluir <ncurses.h>

Adicionar a seguinte linha no final do ficheiro **CMakeLists.txt** target\_link\_libraries(${PROJECT\_NAME} -lncursesw) **Nota**: em “lncursesw” -> “l” não é um 1 e sim um “L” (minúsculo)

Após a alteração do **CMakeLists.txt**, é necessário indicar ao CLion para re-carregar esse ficheiro para considerar as alterações (idem Windows e Mac).



## Windows

Neste sistema a biblioteca ncurses tem o nome de **PDcurses**

1. Instalar as bibliotecas ncurses

Fazer download da biblioteca PDCurses1 e extrair o zip para o disco. Local: **https://github.com/wmcbrine/PDCurses/releases** Os ficheiros extraídos são usados nos passos seguintes.

1 https://github.com/wmcbrine/PDCurses/releases

1. No projeto CLion

É necessário compilar a biblioteca pdcurses para o sistema, e depois copiar o binário obtido para o projeto onde vai ser usada a biblioteca.

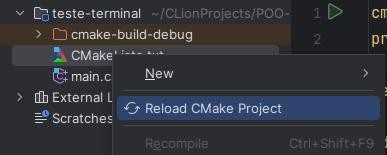
* 1. Abrir o Clion, ir ao menu File > Open e escolher o Makefile na pasta **wincon** da pasta do PDCurses.
  2. Selecionar “Open as Project” 2
  3. Fazer o build. Isto cria o ficheiro **pdcurses.a** na pasta **wincon**.
  4. Copiar o ficheiro **curses.h** da pasta do PDCurses para a raiz do projeto onde se vai usar a biblioteca
  5. Para usar diretamente as funções da biblioteca fazer #include ”curses.h” (com aspas). Usando as classes fornecidas (Terminal.h/cpp) não é necessário fazer este include.
  6. No ficheiro **CMakeLists.txt** do projeto onde vamos usar a biblioteca adicionar (esta entrada não existe no ficheiro) a seguinte linha no final corrigindo o caminho para o ficheiro pdcurses.a:

### target\_link\_libraries(${PROJECT\_NAME} c:/path/pdcurses.a)

**Notas**: 1) **substituir c:/path/** pela diretoria onde está **pdcurses.a**

* + 1. pdcurses.a tem mesmo que ser compilado no sistema windows onde vai ser usado. Compilar este ficheiro e distribuir a colegas poderá não funcionar para esses colegas (depende do sistema deles)
    2. pdcurses.a pode ser colocado dentro do próprio projeto CLion onde é usado. Será apenas necessário configurar o projeto e CLion, mas isso não é abordado aqui.

Após a alteração do **CMakeLists.txt**, é necessário indicar ao CLion para re-carregar esse ficheiro para considerar as alterações (idem Linux e Mac).



2 https://[www.jetbrains.com/help/clion/makefiles-support.html](http://www.jetbrains.com/help/clion/makefiles-support.html)

## Mac-OS

1. Instalar as bibliotecas ncurses

Abrir um terminal e executar o comando

### brew install ncurses

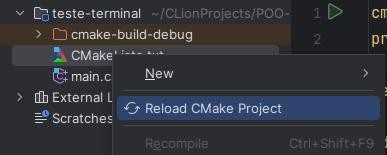
brew é um comando que permite instalar software a partir da linha de comandos em Mac-OS

1. No projeto CLion

Incluir <ncurses.h>

Adicionar a seguinte linha no final do ficheiro **CMakeLists.txt** target\_link\_libraries(${PROJECT\_NAME} -lncurses) **Nota**: em “lncurses” -> “l” não é um 1 e sim um “L” (minúsculo)

Após a alteração do **CMakeLists.txt**, é necessário indicar ao CLion para re-carregar esse ficheiro para considerar as alterações (idem Windows e Linux).



# Uso da funcionalidade da biblioteca

A biblioteca é fornecida com os seguintes ficheiros

* + **Terminal.h** -> Contém a declaração das classes que constituem a biblioteca. É necessário incluir este ficheiro no projeto e que dele se faça *#include* nos ficheiros onde se usa a sua funcionalidade. **É importante analisar este ficheiro para ver o conjunto de funções disponíveis e parâmetros**.
  + **Terminal.cpp** -> contém o código das classes. Não é prioritário ver este código mas tem que ser incluído no projeto.
  + **main.cpp** -> **Exemplo** de utilização. É muito importante que se veja este exemplo para perceber como se usa a biblioteca, mas não deve ser adicionado ao projeto.

Como tarefa inicial, deve ser posto em execução um projeto com os ficheiros da biblioteca tal como fornecidos para:

* + garantir que a configuração no sistema e no CLion quanto a ncurses está a funcionar
  + ver o exemplo a correr e garantir que percebe, pela sua execução e pela análise do código em main.cpp, o que a biblioteca faz e como se usa.

**Funcionalidade e uso**

Existem dois conceitos principais: **Terminal** e **Window**

* + Terminal diz respeito ao ecrã (área visível da consola) - ocupa a área total deste.
  + Window diz respeito a uma parte da área do terminal. É definida com base em coordenadas dentro do terminal, dimensões e pode ter uma moldura opcional. Aquilo que se imprime dentro de uma janela não sai para fora dessa janela e é bastante útil para construir áreas lógicas dentro do ecrã (nesta zona apresenta-se isto, naquela zona apresenta-se aquilo, etc.).

Ambos os conceitos estão encapsulados por classes: Terminal e Window.

* + As classes Terminal e WIndow **estão no namespace term**, o qual deve ser usado da mesma forma que se usa qualquer namespace (exemplo, std).
  + A funcionalidade oferecida pelo Terminal e pela Window é semelhante.
  + A funcionalidade de Terminal e Window é acessível através de objetos destas classes.
  + A maior parte da funcionalidade é invocada usando os **operadores <<** e **>>** em que o objeto

### do lado esquerdo é um objeto de Terminal ou Window.

**Obtenção de objetos Terminal e Window**

Terminal (deve haver apenas um objeto terminal em todo o projeto, naturalmente) Terminal &t = Terminal::instance();

Window (pode haver várias, tem um construtor que define as dimensões) Window window = Window(posicao\_x, posicao\_y, largura, altura);

A funcionalidade de configuração de output (cores, posição do ecrã), e de leitura e escrita será feita usando maioritariamente (mas nem sempre) os **operadores <<** e **>>** direcionados a objetos de Terminal ou Window.

### Nos exemplos seguintes tw é um objeto de Terminal ou e Window

**Definição de cores**

* + As cores devem primeiro ser “inicializadas” com a função

**int init\_color(*número-a-associar*, *cor-de-texto*, *cor-de-background*)**

fica **associada a um número** que será mais tarde usado para referir a cor inicializada.

* + Para usar a cor (previamente inicializada) usa-se

**tw << set\_color( *número-associado-à-cor*)**

### Posicionamento do cursor

* + Usa-se a função

**move\_to(*coluna*, *linha*)**

* + - Início de coordenadas: canto superior esquerdo
    - Leituras e escritas de dados posteriores ocorrem a partir da posição indicada, avançando como habitualmente

### Impressão de dados

* + Usa-se o operador **<<** dirigido a um objeto de Terminal ou Window

### tw << ...

(**...** representa valores e variáveis dos tipos de dados suportados)

* + Tipos de dados suportados
    - string
    - int
    - double
    - char

### Leitura de dados

* + Usa-se o operador **>>** dirigido a um objeto de Terminal ou Window

### tw >> ...

(**...** representa uma referência para um objeto ou variável dos tipos de dados suportados)

* + Tipos de dados suportados
    - String
    - char

### Importante

-> **Leitura de inteiros** (ou outros valores numéricos)

* + - Deve ser lida uma string e depois processada com istringstream para extração dos valores dos tipos de dados pretendidos da forma habitual (obj-de-istringstream >> …).
    - Esta forma de ação tem a ver com o facto de qualquer erro de leitura dever ser processado por quem pretende o valor e não pela biblioteca em si.

### -> Leitura de strings

* + - É lida a string até ao \n, incluindo espaços.
    - Suporta a leitura das teclas especiais de direção (se for a primeira tecla a ser pressionada na string), caso em que a string lida é uma de:

KEY\_UP, KEY\_DOWN, KEY\_RIGHT, KEY\_LEFT, KEY\_RESIZE

Isto é extremamente útil para situações em que se deseje usar essas teclas para movimentar coisas.

### Leitura de caracteres

* + A função **getchar()** permite ler uma tecla sem necessitar de enter no final.

### Outras funcionalidade disponíveis

* + Existem mais funcionalidades de uso menos importante tal como indagar as dimensões do terminal que podem ser descobertas pela análise de Terminal.h

### Adição de funcionalidades e alteração da biblioteca

* + É permitida

**Muito importante**

-> Devem mesmo ser analisados o Terminal.h e o exemplo fornecido.