Wo liegen unter Linux standardmäßig die vorhandenen Header- und Bibliotheksdateien?

Headers:

- aktueller Pfad
- /usr/local/include
- /usr/include
- (/usr/lib/gcc/x86_64-pc-linux-gnu/12.2.0/include)
- (/usr/lib/gcc/x86_64-pc-linux-gnu/12.2.0/include-fixed)

Libraries:

- aktueller Pfad
- /lib
- /usr/lib
- (/usr/local/lib)

Welche Unix-Umgebungsvariablen legen die Pfade fest, in denen gcc nach verfügbaren Header-Dateien und Bibliotheken sucht?

Headers:

- GCC_EXEC_PREFIX
- OBJC_INCLUDE_PATH
- DEPENDENCIES_OUTPUT
- SUNPRO_DEPENDENCIES

Libraries:

- LIBRARY_PATH
- (LD_LIBRARY_PATH)

Erklären Sie die Bedeutung der gcc-Parameter -I, -L, -l

- -I dir, in dem Pfad dir wird nach header files gesucht. Die dabei gewählten Pfaden werden den Systempfaden bevorzugt
- -Ldir, fügt die Pfade von dir hinzu, in welchen nach Libraries gesucht wird
- -l library, der Linker sucht nach der Bibliothek mit dem Namen library

Was ist der Unterschied zwischen #include "header.h" und #include <header.h>?

- Bei #include <header.h> wird nach einem Standard-Header in den Standard-Verzeichnissen mit dem Namen "header.h" gesucht
- Bei #include "header.h" wird in den selbst angegebenen (-I) und dem aktuellen Pfad nach "header.h" gesucht

Erklären Sie den Unterschied zwischen statischem und dynamischem Linken. Wie können Sie statisches Linken erzwingen? Welche Konsequenzen hat das?

Beim statischen Linken wird zur Compile-Time die kompletten Libraries in die fertige Executable kopiert.
Beim dynamischen Linken hingegen wird nur der Name der Library in die Executable kopiert und zur Runtime "ausgewertet".
Somit sorgt statisches Linken dafür, dass bei Änderung der Library, auch das eigene Programm neu kompiliert werden muss, obwohl sich an dessen Code ggf. nichts geändert hab. Außerdem wird dadurch die Executable deutlich größer. Dies könnte aber zu einer besseren Kompatibilität beitragen, da die Systeme nicht die genannten Libraries benötigen, da diese mitgeliefert werden.
Beim dynamischen Linken wird einiges an Speicherplatz eingespart (wenn mehrere Programme die gleiche Library verwenden), da diese nicht jedes mal mit ausgeliefert werden. Ebenso könnte die Library bereits im Arbeitsspeicher geladen sein, falls ein anderes Programm bereits auf diese zugegriffen hat.\

Es gibt mehrere Möglichkeiten statisches Binden zu erzwingen:

- --static an gcc übergeben
- Ist eine Library mit -lXYZ angegeben, so kann man das auf -l:libXYZ.a abändern
- Will man nur lib2 statisch binden: `gcc program.o -llib1 -Wl,-Bstatic -llib2`

Welche Informationen liefert Ihnen das Linux-Tool ldd?

ldd zeigt die Dependencies, also shared libraries, des Programmes an und wo diese auf dem lokalen System zu finden sind, bzw. auf welche Libraries das Programm an bei der Ausführung zugreift.
.so steht dabei für 'shared object'

```
ldd /bin/alacritty
    linux-vdso.so.1 (0x00007fffbdcbf000)
    libxcb.so.1 => /usr/lib/libxcb.so.1 (0x00007f89d0b4c000)
    libfreetype.so.6 => /usr/lib/libfreetype.so.6 (0x00007f89d0a7d000)
    libfontconfig.so.1 => /usr/lib/libfontconfig.so.1 (0x00007f89d0a2d000)
    libgcc_s.so.1 => /usr/lib/libgcc_s.so.1 (0x00007f89d03e0000)
    libm.so.6 => /usr/lib/libm.so.6 (0x00007f89d02f8000)
    libc.so.6 => /usr/lib/libc.so.6 (0x00007f89d0111000)
    libXau.so.6 => /usr/lib/libXau.so.6 (0x00007f89d0a26000)
    libXdmcp.so.6 => /usr/lib/libXdmcp.so.6 (0x00007f89d0109000)
    libz.so.1 => /usr/lib/libz.so.1 (0x00007f89d00ef000)
    libbz2.so.1.0 => /usr/lib/libbz2.so.1.0 (0x00007f89d00dc000)
    libpng16.so.16 => /usr/lib/libpng16.so.16 (0x00007f89d00a3000)
    libharfbuzz.so.0 => /usr/lib/libharfbuzz.so.0 (0x00007f89cffb7000)
    libbrotlidec.so.1 => /usr/lib/libbrotlidec.so.1 (0x00007f89cffa9000)
    libexpat.so.1 => /usr/lib/libexpat.so.1 (0x00007f89cff7e000)
    /lib64/ld-linux-x86-64.so.2 => /usr/lib64/ld-linux-x86-64.so.2
(0x00007f89d0bbf000)
    libgraphite2.so.3 => /usr/lib/libgraphite2.so.3 (0x00007f89cff5c000)
    libglib-2.0.so.0 => /usr/lib/libglib-2.0.so.0 (0x00007f89cfe1f000)
    libbrotlicommon.so.1 => /usr/lib/libbrotlicommon.so.1 (0x00007f89cfdfc000)
    libpcre2-8.so.0 => /usr/lib/libpcre2-8.so.0 (0x00007f89cfd61000)
```