Curs d'Introducció a LaTeX

Optica't UAB

Versió 1.0

Benvolguts i benvolgudes a aquest text introductori de LaTeX. Som Optica't, un grup de divulgació d'òptica física format per estudiants de la Universitat Autònoma de Barcelona. Aquest text està pensat per a facilitar l'aprenentatge d'aquesta eina a alumnes que comencin els seus estudis universitaris. Clarament, està orientat a graus científics, el nostre aprenentatge de LaTeX es basa en les necessitats i la curiositat que hem desenvolupat al llarg dels cursos en carreres com física o nanociències. Tot i això, pensem que LaTeX és una eina molt útil per a més disciplines i esperem que molta gent pugui fer profit d'aquest curs.

Índex

Agraïments

Capítol 1

Introducció i conceptes bàsics

Dediquem aquest capítol a explicar des de zero com funciona LaTeX i com podem començar a crear els nostres primers documents. La finalitat del capítol és que us quedeu amb els conceptes bàsics de com funciona l'eina, la seva sintaxi, els possibles errors que trobareu, etc.

1.1 Què és LaTeX i la seva utilitat

LaTeX és un editor de textos que funciona de forma similar a un llenguatge de programació compilat. Està basat en el llenguatge de baix nivell TEX, i està pensat principalment per a escriure textos científics de forma senzilla i elegant. La filosofia d'aquesta eina és que qui redacti no es preocupi del format del document, només en què ha d'escriure. A diferència d'un editor de textos WYSIWYG (en anglès, 'el que veus és el que obtens') al treballar amb LaTeX no podem veure el resultat del que redactem al moment d'escriure-ho. En LaTeX es treballa directament amb codi i després un compilador ho interpreta i genera un resultat (per exemple imprimeix el document per pantalla o genera un arxiu `.pdf`).

En un principi pot semblar que escriure en codi i no veure què és el que estem generant és contraproduent i pot fer menys atractiva la idea de treballar amb aquest editor i no amb altres més populars. Però, com veureu al llarg d'aquest text, aquest desavantatge queda opacat per la quantitat d'eines útils i de personalització que s'obtenen en treballar amb LaTeX.

Aquest curs se centra a abastar tot el necessari per a començar a escriure textos científics, com ara lliuraments de diverses assignatures de la carrera o per al treball de fi de grau. Però, en ser només un text introductori, no podrem explicar totes les eines que ofereix LaTeX bàsic i encara menys totes les eines que ha creat la comunitat. És necessari dir també que la utilitat d'aquest editor de textos va més enllà dels textos científics semblants a *papers*. LaTeX té incorporades eines per a escriure llibres, per a escriure música, per a fer presentacions... Us convidem a informar-vos més enllà d'aquest text perquè podeu jugar amb totes les possibilitats que ofereix LaTeX tot i que no estudieu a cap branca científica.

1.2 Entorns per a treballar amb LaTeX

Per començar a treballar amb LaTeX primer necessitem saber on! És a dir, necessitem saber quin és el nostre entorn de treball. Igual que passa amb els llenguatges de programació, podem córrer LaTeX a dins d'una gran varietat d'entorns. Cada entorn és lleugerament diferent de l'altre, les seves característiques depenen de la versió i de quin sigui el flux de treball pensat específicament per a aquell entorn. Escollir un bon entorn pot arribar a ser una mica enrevessat, per tant, per a facilitar la tasca a les persones que encara no han fet servir LaTeX o que no estiguin familiaritzades en treballar amb programes de manera local centrarem el curs en l'entorn d'Overleaf. Sense ser exhaustius també parlarem sobre altres entorns com algunes eines d'escriptori o a escriure directament en un document de text amb eines més específiques de programació.

Per al lector o lectora que ja tingui experiència amb Overleaf i vulgui fer servir les altres eines, pròximament crearem una guia que complementi aquest curs amb més informació, de moment ens centrarem en que pugueu escriure el vostre primer document en LaTeX.

1.2.1 Overleaf

L'entorn per excel·lència per a aprendre LaTeX i per a fer petits projectes col·laboratius és Overleaf. Per a fer-ho servir no cal instal·lar-se res, només necessites connexió a internet i un usuari. Overleaf és un entorn de LaTeX en línia pensat per a fer més fàcil la col·laboració entre usuaris a l'hora de crear un document compartit. Els seus avantatges són:

- No requereix cap instal·lació, només tenir un compte. Per a poder accedir a un compte d'Overleaf només necessiteu un correu electrònic i entrar a la web oficial[1]
- Interpreta el codi fins i tot si aquest conté algun petit error
- Permet treballar conjuntament amb diverses persones[2]
- Permet classificar els teus projectes amb etiquetes
- Té instal·lats alguns paquets que el LaTeX pur no té per defecte
- Té un sistema de navegació i una visualització del log de compilació intuïtius
- Té integrades algunes eines que faciliten la generació de codi, com ara botons o *short cuts* que canvien el format del text com en Word o un generador de taules buides
- Proporciona un editor visual on es pot treballar en un entorn semblant a un editor WYSIWYG

Com aquesta serà l'eina que farem servir a la resta del curs, explicarem una mica més sobre com funciona. Un cop tinguem obert un compte podrem crear un nou projecte des del menú principal. Overleaf ofereix moltes plantilles preestablertes, però per a aprendre treballarem amb una plantilla buida. Un cop obert el nou projecte en blanc veurem la següent pantalla:

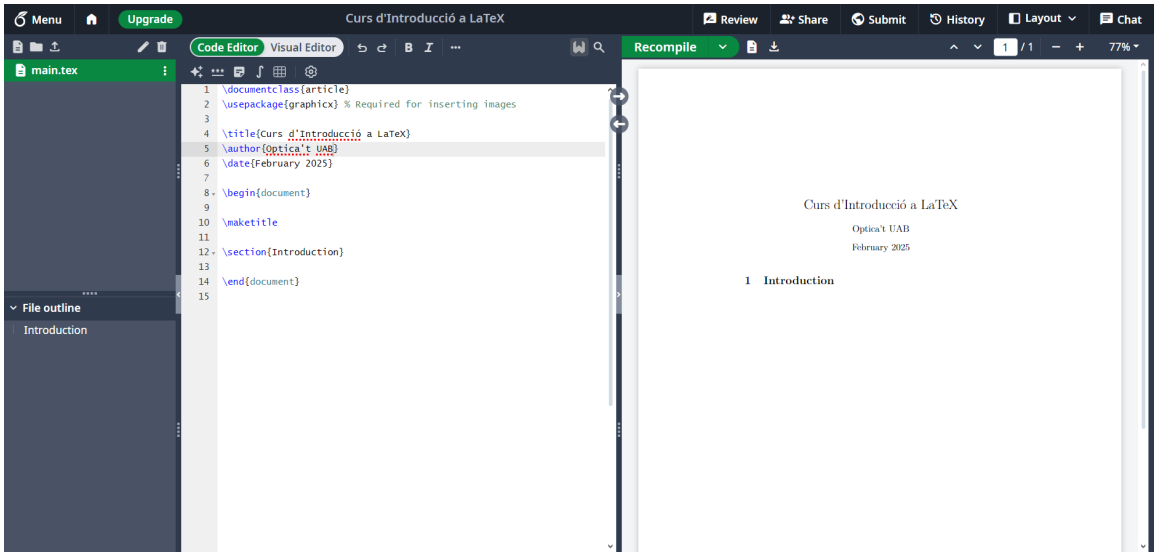


Figura 1.1: Entorn de treball d'Overleaf per a un projecte en blanc.

Observem de la Figura 1.1 que l'entorn se separa en tres parts. A l'esquerra del tot observem un menú, l'explorador d'arxius i de projecte. Aquest explorador funciona com a un gestor de carpetes i d'arxius, molt similar a l'explorador d'arxius de Windows. A la part superior de l'explorador aniran apareixent els arxius que fem servir al projecte. Podem mantenir un ordre d'aquests arxius utilitzant carpetes i canviant el nom de cada arxiu com vulguem. A la part inferior de l'explorador apareixerà un llistat de títols que ens serveixen per a moure'ns pel projecte de forma més senzilla, expliquem el seu funcionament amb més detall a la secció 2.5.2.

A la dreta de la pantalla s'observa un visor pdf. Aquest visor permet veure per pantalla com va quedant

[1] Us deixem l'enllaç vigent actualment: https://es.overleaf.com

[2] Actualment (inici del 2025) la limitació per a editar el mateix projecte d'Overleaf es limita a dos usuaris. La resta d'usuaris que entrin al projecte només poden llegir-ho.

el document que estem generant amb el codi que hem escrit. Notareu que a la part superior esquerra del visor hi ha un botó verd on posa `Recompile`. Cada cop que vulguem actualitzar el pdf amb la informació que hem escrit en codi hem de fer click a `Recompile`[3]. Diem a aquesta acció recompilar o compilar el codi.

Entre l'explorador i el visor tenim l'editor de text. Aquest editor és igual que qualsevol altre en programació. Allà és on escrivim totes les línies de codi necessàries per a generar el nostre document. Tot i això, aquesta finestra intermitja no sempre farà d'editor. Overleaf permet treballar amb més tipus d'arxiu que els de LateX com per exemple imatges. Per tant és més encertat pensar en l'editor com un entorn dinàmic de treball i no només com un simple editor de text.

Es pot configurar la visibilitat tant de l'editor com del visor pdf a partir de les opciones del desplegable `Layout`. Aquestes opcions deixen només visible un dels dos, els dos o permet separar-los en finestres diferents del navegador (això és útil si per exemple treballem a doble pantalla).

1.2.2 LaTeX d'escriptori

Aquí va el text que redactarà Aurem :)

1.2.3 Eines alternatives

Per a la gent més experimentada o amb més interés en la programació deixem el següent entorn que, tot i potser semblar més complicat de fer servir, permet personalitzar completament el teu entorn de treball. LaTeX és un llenguatge, no un programa, per tant desde qualsevol editor de text podem escriure en LaTeX i, fent servir el compilador corresponent podrem generar la sortida que vulguem. Ara bé, existeixen eines que faciliten aquesta feina i ens permeten configurar el nostre entorn de forma més visual. Podriem fer una llista enorme amb eines de visualització útils i personalitzables, però una de les més populars (i des de la que s'està redactant aquest text) és Visual Studio Code. La instalació i configuració d'aquest entorn queda fora dels proposits d'aquest curs, però us convidem a informar-vos una mica sobre aquesta eina ja que no serveix només per a LaTeX. Visual Studio Code permet treballar amb LaTeX (i molts més llenguatges) en local, té integrat un explorador de directoris molt visual (no cal barallar-se amb la bash per a navegar per les carpetes) i permet afegir extensions pensades per a treballar amb LaTeX.

1.3 Estructura bàsica d'un document

De forma similar a un llenguatge de programació quan escrivim un document en LaTeX necessitem seguir un cert ordre i respetar una sintaxi concreta per a que el compilador pugui comprendre i traduïr correctament el nostre document. Aquestes normes que s'han de seguir no són complicades, de fet amb Overleaf són similars a escriure en Word. Com hem comentat abans, LaTeX està pensat per a que no ens haguem de preocupar per la forma sinó més per qué escrivim, per tant és aquí on hi ha la diferència principal amb altres eines d'edició de textos.

1.3.1 Sintaxi

L'estructura d'un document de LaTeX se separa en dues parts ben diferenciades:

- **Preàmbul**: a on escrivim la "configuració" del document
- **Text principal**: a on escrivim el cos del document

Separem aquestes dues parts de forma explícita, reservant sempre les primeres línies de codi per al preàmbul i la resta per al text principal. Veiem com a exemple el codi que escriu Overleaf per defecte quan obrim un document en blanc:

Exemple 1.1 [4]

[3] No cal fer click sempre, existeixen dues alternatives. La primera és fer servir *short cuts*, és a dir, prémer `Ctrl + s` o bé `Ctrl + Enter`. La segona és activar la funció automàtica de recompilat. Podeu accedir a aquesta opció desde el botó `Recompile`, prement la fletxa que té al costat.

[4] Tant Overleaf com LaTeX permet fer servir accents escrits directament amb el teclat o amb codi (ho veurem més endavant). Però, l'entorn que fem servir per a escriure els exemples no els detecta correctament. Això no serà un problema per a volsaltres a l'hora de redactar un document sense aquest recurs. Si us plau, feu servir accents.

```
\documentclass{article}
\usepackage{graphicx} % Required for inserting images

\title{Curs d'Introduccio a LaTeX}
\author{Optica't UAB}
\date{February 2025}

\begin{document}

\maketitle

\section{Introduction}

\end{document}
```

Des de la primera línia fins a la setena és el preàmbul. La resta és el text principal i comença amb el comandament \begin{document}. L'exemple 1.1 ens permet veure els tres tipus de comandaments bàsics del preàmbul i com s'estructura un document. El preàmbul de l'exemple comença declarant a la primera línia el tipus de document que estem redactant. Seguidament, a la segona, declara el paquet `graphicx`. Els últims tres comandaments configuren tres objectes predefinits (el títol, l'autor i la data). Veiem doncs que al preàmbul els comandaments bàsics que podem fer servir són:

- Declaració de tipus de document
- Declaració de paquets
- Configuració d'entorns pre-establerts

Parlarem amb més detall d'aquests comandaments a les següents seccions (1.3.2, 1.3.3 i 1.3.4). La resta del document, alló que es plasmarà al compilar, es redacta en un entorn delimitat pels comandaments \begin{document} i \end{document}. El codi que hi ha escrit entre aquests delimitadors serveix per a imprimir al document el títol i el títol de la secció anomenada "Introduction". Més endavant ens interessarem per aquests comandaments. Encara més important per a aquesta part del codi és com es redacta text pla per a que s'imprimeixi com volem per pantalla. Veiem-ne alguns exemples:

Exemple 1.2

```
\documentclass{article}

\begin{document}
    Aixo s'escriu en text pla, amb una mida de font standar i amb
    uns marges per defecte.
\end{document}
```

Sortida:

Aixo s'escriu en text pla, amb una mida de font standar i amb uns marges per defecte.

Com veiem el text s'escriu justificat a l'amplada del document i amb una mida de text normal. La mida i el tippus de lletra es pot modificar tant a la configuració del preàmbul com al mateix codi (veure secció 2.1). Si tenim més text aquest s'adaptarà en forma de paràgraf. Existeixen diverses formes de separar dos paràgrafs i cada forma crea un salt de línia de diferent mida i genera o no indentació per defecte.

Exemple 1.3

```
\documentclass{article}

\begin{document}
    Aquest es un text d'exemple que omplira l'amplada del document
    formant un unic paragraf. Podem separar-ho en altres paragrafs de
    diferents maneres.\\
    Aquest paragraf esta separat per $\backslash\backslash$. S'aplica un
     salt de linia normal, amb una separacio igual a la d'un salt de
    linia del mateix paragraf i no aplica indentacio.

```

```
7       Deixar una linia en blanc tambe genera un altre paragraf com al
        escriure $\backslash\backslash$ pero a mes aplica indentacio. Si
        LaTeX detecta que hi ha mes d'un espai en blanc normalment avisa amb
         un error de sintaxi. Overleaf permet fer aquest espais en blanc i
        ho interpreta com a un de sol.Podem generar el mateix efecte
        separant el paragraf amb el comandament $\backslash$par. LaTeX quan
        es troba aquest comandament enten que s'ha acabat un paragraf i que
        lo que ve a continuacio pertany a un seguent paragraf. \par Per
        exemple aquest text esta separat pel comantament $\backslash$par
        .\\\\
8       Aquest darrer paragraf se separa amb quatre $\backslash$. Aquesta
        forma de separar paragrafs no genera indentacio i crea un salt de
        linia mes gran, fent que el text se separi en dos de forma mes
        visual.
9   \end{document}
```

Sortida:

Aquest es un text d'exemple que omplira l'amplada del document formant un unic paragraf. Podem separar-ho en altres paragrafs de diferents maneres.
Aquest paragraf esta separat per \\. S'aplica un salt de linia normal, amb una separacio igual a la d'un salt de linia del mateix paragraf i no aplica indentacio.

Deixar una linia en blanc tambe genera un altre paragraf com al escriure \\ pero a mes aplica indentacio. Si LaTeX detecta que hi ha mes d'un espai en blanc normalment avisa amb un error de sintaxi. Overleaf permet fer aquest espais en blanc i ho interpreta com a un de sol.Podem generar el mateix efecte separant el paragraf amb el comandament \par. LaTeX quan es troba aquest comandament enten que s'ha acabat un paragraf i que lo que ve a continuacio pertany a un seguent paragraf.

Per exemple aquest text esta separat pel comantament \par.

Aquest darrer paragraf se separa amb quatre \. Aquesta forma de separar paragrafs no genera indentacio i crea un salt de linia mes gran, fent que el text se separi en dos de forma mes visual.

El detall de com escriure \ dins del codi per a que s'imprimeixi correctament ho explicarem més endavant. A l'exemple 1.3 podem veure quatre formes de separar paràgrafs. L'ús d'un salt o altre ve a gust de qui està redactant el document. En aquest text, per exemple, normalment separem els paragrafs amb \\\\ per a obtenir un efecte visual més polit. La separació de paràgrafs de vegades ve condicionada pel que té el paràgraf al seu voltant, usualment pel codi just a sobre d'ell[5]. En cas de voler afegir o treure una indentació desitjada o no podem fer servir el comandament `\indent` o `\noindent`.

La indentació a LaTeX pot portar una mica de confusió no només amb el que s'imprimeix en pantalla al genrar el resultat, sinó també amb la indentació dins del propi codi. L'estructura del codi que escrivim ve imposada pel fet que hem de fer entendre al compilador què volem que surti exactament. Però, per a poder facilitar la nostra lectura del codi podem fer servir una jerarquia amb la indentació dins del codi. Fixem-nos novament en l'exemple 1.3, el text que s'acabarà imprimint al document es troba indentat respecte als delimitadors del document (el `begin` i el `end`). Podriem escriure aquest codi sense aplicar aquesta indentació i el resultat per pantalla acabaria sent el mateix que abans. Aquesta indentació global el compilador la ignora, però a nosaltres ens és útil per a saber que aquella part del codi pertany al document. En cas de ternir més entorns a dins del codi (com veurem a la secció 1.3.4) podem aprofitar aquesta indentació per a poder escriure un codi més polit que pugui ser fàcilment entés tant pel compilador com per a qui escriu o revisa el codi.

Paraules reservades i comandaments

Com heu pogut observar als exemples que hem posat fins ara hi ha algunes paraules que les hem posat en color blau. Semblant a un llenguatge de programació, LaTeX té un conjunt de paraules i simbols

[5] Això ho veureu a mesura que escriviu documents en LaTeX, lo més usual és veure-ho quan es comença un paràgraf després de posar una figura o una taula. Per exemple, al finalitzar l'exemple 1.3 comença un paràgraf amb indentació automàtica.

reservats per a que el compilador entengui si estem escrivint text pla o si estem donant una ordre concreta. Els comandaments (ja siguin del preàmbul o a dins del text o entorns matemàtics) venen declarats començant pel símbol "\"[6]. Entorns visuals com Overleaf faciliten la tasca de saber quines són aquestes paraules reservades canviant el seu color al codi o fins i tot fent prediccions del codi que volem escriure.

No és viable aprendre de forma activa quines són aquestes paraules (no recomenem provar-ho, és una tasca molt farragosa). Lo més útil és practicar i amb el temps acostumar-se a la forma intuituva que té LaTeX de seleccionar aquestes paraules, ja que normalment són instruccions literals en anglès. Al llarg del curs explicarem algunes d'aquestes paraules reservades i a l'Annex deixarem algunes taules amb exemples útils que podeu fer servir al vostre dia a dia.

A més, veureu que acompanyant a aquestes paraules reservades normalment apareixen { } o [ ]. Entre claus normalment s'escriu l'argument de la funció que estiguem cridant amb la paraula reservada. Entre claudators normalment s'escriuen les opcions de configuració de la funció.

1.3.2 Tipus de document

Fixem-nos ara en la primera línia del preàmbul. Com hem dit aquest comandament declara quin tipus de document estem escrivint. Segons com configurem aquesta línia, el text que redactem sortirà amb una configuració predefinida o altre. Aquesta configuració és molt bàsica, però pot portar a problemes que semblen que no tenen solució si no es té en compte qué hem configurat i que no. Existeixen molts tipus de documents que enten LaTeX, comentem per sobre els més comuns:

- `article` - Aquest és el tipus de document que normalment fareu servir a la carrera. Està pensat per a escriure documents científics curts, de forma que permet estructurar el text en seccions.
- `report` - En cas de necessitar fer un article més llarg podeu fer servir aquest tipus de document. Un `report` permet a més de seccions separar el text en capítols.
- `book` - Aquest és el tipus de document que fem servir per a redactar aquest curs. Treballa de forma similar a un report (és un text llarg i amb capítols) pero afegeix detalls necessaris per a estructurar un llibre. Un exemple molt clar d'això és que de forma automàtica genera l'espai en blanc adient per a que els capítols comencin en una pàgina imparella.
- `letter` - En cas de voler escriure una carta amb una estructura formal podeu fer servir aquesta classe de document. `letter` permet declarar al preàmbul algunes variables predefinides (com la signatura o l'acomiadament) que serveixen per a estructurar la carta de forma molt senzilla.
- `beamer` - Aquest darrer format serveix per a crear presentacions. És una classe de document molt personalitzable, però és més complicada de fer servir i això la fa una mica limitada per a la gent que comença a fer servir LaTeX.

Cada classe de document es pot configurar com més ens convingui. L'estructura del comandament sencer és `\documentclass[...]{`*`class`*`}` on entre claudators podem escriure la configuració i entre claus la classe de document que vulguem. Les configuracions poden ser molt variades i es pot posar més d'una entre els claudators sempre que les separem per comes. Deixem algunes configuracions comunes amb una breu explicació i alguns exemples:

- Es pot modificar la mida de la lletra del text pla general posant `10pm`, `11pm`, `12pm`, etc. La unitat de mesura `pm` s'anomena punt i correspon a aproximadament 0,35 mm al paper.
- Podem configurar la mida del paper segons desitjem. Per defecte en un document `book` la mida és A4, però en altres pot variar. Podem explicitar diferents mides com `a4paper` (mida A4), `letterpaper` (mida de carta, 216 × 279 mm), `b5paper` (format B5, 176 × 250 mm), etc.
- Configurem el format d'impressió a una o doble cara, amb els comandaments `oneside` i `twoside` respectivament. Aquesta configuració canvia per exemple quan comença un capítol, si a la següent pàgina o a la següent pàgina imparella. Cada configuració està pensada per a que es pugui enquadernar a una o a dues cares.

[6] És per aquest motiu que no hem escrit directament el símbol al codi per a que s'imprimeixi en la sortida del document, ja que LaTeX ho interpretaria com a un comandament i no com a un símbol

- Configurem la distribució del text ja sigui a una o a doble columna[7] amb els comandaments `onecolumn` per a una columna i `twocolumn` per a dues.
- Podem escollir si els capítols comencen a qualsevol pàgina (`openany`) o a les pàgines imparelles (`openright`).

Exemple 1.4 Article en format A4 amb lletra de mida 10pm:

```
\documentclass[10pm]{article}
```

Exemple 1.5 Report mida A4, a dues columnes, lletra mida 12pm i a una cara:

```
\documentclass[a4paper,12pm,oneside,twocolumn]{report}
```

Exemple 1.6 Configuració d'aquest document:

```
\documentclass[12pm,twosides,onecolumn,openany]{book}
```

Veiem a l'exemple 1.6 que hem aplicat `openany` i no hem explicitat `a4paper` (no és necessari, `book` ho té onfigurat per defecte). Això es deu a que aquest document està escrit per a llegir-se en format digital i no en paper. La versió en paper d'aquest document no hauria d'escriure l'opció `openany` i deixar per defecte `openright` (es podria escriure explícitament, però `book` té aquesta darrera opció per defecte).

1.3.3 Paquets

Com hem dit existeixen molts comandaments que podem escriure a partir de paraules reservades. Amb aquests comandaments podem, com si fos un llenguatge de programació, configurar funcions que ens modifiquen el que imprimim per defecte en un document escrit en LateX. Però, aquesta tasca és molt farragosa, ja que per a fer servir eines molt comunes hauriem de saber la configuració exacta de cada detall que necessitem escriure. És per això que, novament com als llenguatges de programació, existeixen els paquets. Aquests paquets són codi en LaTeX escrit per a configurar diverses funcions que necessitem per a redactar, hi ha molts i de molt diferents. Alguns venen per defecte amb LaTeX i altres venen per part de la comunitat. El funcionament de cada paquet és molt diferent a cada un, fins i tot podem trobar alguns obsolets o incompatibles entre ells. Però, de forma senzilla podem explicar el seu funcionament dient que són comandaments que indiquen a LaTeX que existeixen noves paraules reservades preconfigurades.

Per a poder fer servir un paquet hem de cridar-lo explícitament al preàmbul amb el comandament `\usepackage{...}`, just després de declarar el tipus de document. Podeu trobar paquets molt interessants buscant per internet, us deixem un breu llistat amb alguns exemples:

- `\usepackage[a4paper]{geometry}` - Aquest paquet és útil per a configurar els marges del document, aquest exemple adapta els d'un de mida A4.
- `\usepackage{caption}` - Permet configurar la lletra dels títols de taula i peus de figura.
- `\usepackage{fancyhdr}` - Serveix per a configurar la capçalera i els peus de pàgina.
- `\usepackage{mhchem}` - Permet escriure més fàcilment fòrmules i reaccions químiques.

Fent referència a l'explicació simplificada de lo que és un paquet podem veure l'exemple de `mhchem`. Aquest paquet indica a LaTeX que existeix una nova paraula reservada assignada al comandament "`\ce{...}`" la qual amb els arguments adients ens permet escriure fòrmules o reaccions químiques més fàcilment.

Un detall important a l'hora d'utilitzar paquets és que no tots es troben instal·lats per defecte a LaTeX. Això depèn molt de l'entorn on es treballi, pot variar fins i tot segons la versió o la distribució que us hagueu instal·lat en local. Overleaf té l'avantatge que la majoria de paquets útils ja venen instal·lats i no us haureu de preocupar per això[8]. La resta d'usuaris que no feu servir Overleaf haureu d'instal·lar els paquets que no tingueu per defecte segons quin entorn estigueu fent servir, ja sigui des de la web dels creadors del paquet o directament llençant un comandament des de la bash en Windows o Linux.

[7] Això configura el text de forma general, tot el text s'estructurarà en una a dues columnes. Per a obtenir una millor personalització de la doble columna farem servir altres comandaments (veure secció 2.4).

[8] Per exemple, `mhchem` es pot fer servir a Overleaf sense cap problema, però en local s'ha d'instal·lar prèviament.

El funcionament de cada paquet es pot trobar a internet, normalment es pot trobar a la bibliografia oficial del paquet o es pot demanar ajuda a eines d'intel·ligència artificial per a que us donin alguna breu explicació. En aquest curs farem servir i explicarem com funcionen diversos paquets comuns o útils.

1.3.4 Entorns

Posem novament enfasi a les paraules reservades. Com hem dit anteriorment el codi que s'acabarà imprimint al document es troba entre els delimitadors `\begin{document}` i `\end{document}`. Les paraules clau `begin` i `end` són dues paraules reservades molt comunes en un codi de LaTeX, ja que obren i tanquen entorns específics dins del codi. Els arguments que accepten aquests delimitadors poden ser molt diversos, de fet podem obtenir nous arguments afegint paquets. L'ús de delimitadors per a construïr entorns és molt útil tant per a accedir a les funcions que necessitem per a escriure el nostre text com per a seguir una jerarquia al codi (com vam dir al parlar de la sintaxi). Ja sigui indentat o no, el codi que es troba entre els delimitadors constitueix un bloc aïllat de la resta del codi. Qualsevol comandament (llevat d'aquells que actuen globalment) que escrivim dins de l'entorn delimitat només afectarà al codi que hi hagi dins.

Exemples d'entorns propis de LaTeX són l'entorn matemàtic (parlem més profundament al capítol 3), `equation`; l'entorn de justificació central, `center`; l'entorn de justificació cap a la dreta, `flushright`; l'entorn de les figures, `figure`; etc. Exemples d'entorns d'alguns paquets són: ambl el paquet `subcaption` l'entorn de les subfigures, `subfigure`; amb el paquet `tcolorbox` l'entorn d'un quadre de text amb marcs i fons configurable, `tcolorbox`; amb el paquet `listings` l'entorn que permet escriure amb diferent format que el text normal codi en altre llenguatge de programació, `lstlisting`, etc. Aquests últims exemples necessiten primer una configuració extra al preàmbul, per tant són una mica més avançats que la resta d'entorns que venen per defecte amb LaTeX. Podeu veure com es fan servir aquests entorns al següent exemple:

Exemple 1.7

```
\documentclass{article}

\usepackage{subcaption}
\usepackage{listings}
\usepackage{tcolorbox}

\begin{document}
    Aquest text esta justificat per defecte ja que no es troba dins de
    cap entorn, nomes es troba dins del document.\\

    \begin{center}
        El text que es troba aqui esta justificat al centre. S'observa
        com per a poder veure millor el codi fem servir indentacio, pero
         com hem dit el compilador omet aquesta indentacio global. Tambe
         es veu com separem en paragrafs. La combinacio d'espai en blanc
         i $\backslash\backslash$ es equivalent a quatre $\backslash$.
    \end{center}

    \noindent Parlarem mes endavant, pero podeu veure un exemple
    matematic:

    \begin{equation*}
        a^2 + b^2 = c^2
    \end{equation*}

    \begin{flushright}
        Podem tambe escriure text justificat cap a la dreta. Qualsevol
        objecte dins d'aquet entorn que no tingui una justificacio fixa
        es justificara cap a la dreta juntament amb el text. Un exemple
        d'aixo pot ser una imatge.
    \end{flushright}

```

```
24        \begin{tcolorbox}[colframe=red, colback=black!5!white, sharp corners
          , width=0.85\textwidth,boxrule=0.2mm,parbox=false]
25            Aquest text te justificacio per defecte pero envoltat per una
              caixa. La configuracio explicita dels colors i la mida es troba
              entre claudators. La configuracio implicita es troba en el
              preambul del document del curs, parlarem mes endavant d'allo.
26        \end{tcolorbox}
27    \end{document}
```

Sortida[9]**:**

Aquest text esta justificat per defecte ja que no es troba dins de cap entorn, nomes es troba dins del document.

El text que es troba aqui esta justificat al centre. S'observa com per a poder veure millor el codi fem servir indentacio, pero com hem dit el compilador omet aquesta indentacio global. Tambe es veu com separem en paragrafs. La combinacio d'espai en blanc i \\ es equivalent a quatre \.

Parlarem mes endavant, pero podeu veure un exemple matematic:

$$a^2 + b^2 = c^2$$

Podem tambe escriure text justificat cap a la dreta. Qualsevol objecte dins d'aquet entorn que no tingui una justificacio fixa es justificara cap a la dreta juntament amb el text. Un exemple d'aixo pot ser una imatge.

Aquest text te justificacio per defecte pero envoltat per una caixa. La configuracio explicita dels colors i la mida es troba entre claudators. La configuracio implicita es troba en el preambul del document del curs, parlarem mes endavant d'allo.

Veiem a l'exemple 1.7 com fem servir alguns dels entorns que hem mencionat abans. Els detalls del funcionament de cada un ho veurem després, de moment quedeu-vos amb com es treballa amb els entorns.

Aquesta forma de treballar és essencial, ja que ens permet fer servir comandaments locals que afecten a tot el codi que es trobi dins d'un entorn. Per exemple, existeix el comandament `centering`. Aquest és un comandament local, que afecta a tot el seu entorn fins a trobar uns delimitadors (si no hi ha un entorn escrit, els delimitadors són els del mateix `document`). El comandament `centering` justifica al centre tot alló que es trobi al seu voltant fins a trobar-se uns delimitadors. Si és text lo que hi ha dins d'aquest entorn, `centering` funciona igual que l'entorn `center`.

[9] Un detall que no és rellevant però que si que podreu notar en cas d'haver compilat aquest mateix codi o de similars és que LaTeX ens avisa que tenim un paquet carregat però que no fem servir, el paquet `subcaption`. Als exemples no mostrarem aquests avisos, ja que seria ficar-nos a explicar algunes sortides poc rellevants del log de LaTeX i al cap i a la fi no acaba afectant al document final. Aquesta notificació es marca com a error de sintaxi, però el compilador igualment genera el codi resultant ignorant l'error.

Capítol 2

Format de text i estructura

Aquest capítol del curs el dedicarem a aprendre a construir documents de LaTeX amb les eines més senzilles d'edició del format del text que en permet LaTeX pur. Veureu que els diferents apartats són, bàsicament, una recopilació dels comandamens més comuns que es fans servir per a redactar la majoria de textos científics amb aquesta eina.

2.1 Format del text pla

Al capítol anterior hem vist com funciona la sintaxi del text pla, com podem separar-lo per paràgrafs i com fer servir paquets i entorns útils que ens són dmolt bons per a poder redactar documents interessants. Ara parlarem de certs comandaments, també molt bàsics, que permeten configurar el tipus de lletra que fem servir.

Hi ha molts comandaments, ja siguin integrats a LaTeX o que provenen de paquets que ens modifiquen la font del text pla que redactem. Tots aquests comandaments funcionen de la mateixa forma: el text afectat ha de ser l'argument del comandament. Podem fer una breu classificació d'aquests comandaments amb alguns exemples que us deixem a continuació.

Modificacions comunes

\textbf{Text en negreta} **Text en negreta** \textit{Text en cursiva} *Text en cursiva*

\underline{Text subratllat} Text subratllat

Tipus de font

\text{Text normal}[1] Text normal \textsf{Text en Sans Serif} Text en Sans Serif

\texttt{Text mecanografiat} `Text mecanografiat`

\textsc{Text en majúscules petites} Text en majúscules petites

Per a fer canvis en la mida de la lletra fem servir uns comandaments locals que afecten tot l'entorn on treballem. Tan com si escribim en un entorn concret d'alguna funció com si escrivim text pla podem delimitar la part del codi a la que afecten aquests comandaments fent servir claus com a delimitadors. Un exemple general (i inventat) d'aquesta sintaxi per als comandaments locals és:

{\comandament Aquest text està afectat pel comandament exemple}.

Posem a continuació alguns exemples de mida de lletra.

[1] Aquest comandament i un altre ens serà útil en entorns matemàtics. Per escriure text pla normal no cal fer-lo servir.

Mida de lletra

{\tiny Text tiny} Text tiny {\footnotesize Text de footnote} Text de footnote

{\large Text large} Text large {\huge Text huge} Text huge

Podeu trobar més exemples i el llistat complet de mides a l'Annex. Naturalment aquestes modificacions de les fonts es poden convinar convenientment, deixem també alguns exemples.

Exemple 2.1

```
\documentclass{article}

\begin{document}
    \texttt{\LARGE Text mecanografiat i de mida LARGE}\\\\
    \textsc{\textit{Text amb majuscules i en cursiva}}\\\\
    \textsf{\textbf{\footnotesize Text en Sans Serif en negreta i de
    mida footnotesize}}
\end{document}
```

Sortida:

Text mecanografiat i de mida LARGE

Text subratllat i en cursiva

Text en Sans Serif en negreta i de mida footnotesize

Però, s'ha d'anar amb cura ja que existeixen alguns comandaments incompatibles entre ells. Normalment quan això passa LaTeX avisa com un error de sintaxi. El document normalment compilarà, però escollirà de forma jeràrquica quin comandament afectarà a la lletra o quin no. Posem'ne un exemple:

Exemple 2.2

```
\documentclas{article}

\begin{document}
    \textsc{\textit{\footnotesize Text en cursiva i en majuscules
    petites}}\\\\
    \textit{\textsc{\Large Text en majuscules petites i en cursiva}}
\end{document}
```

Sortida:

Text en cursiva i en majuscules petites

TEXT EN MAJUSCULES PETITES I EN CURSIVA

Veiem que els comandaments \textit i \textsc són incompatibles. La jerarquia que se segueix al codi és de dins cap a fora, és a dir, domina el primer comandament que selecciona el text amb les claus. La resta de comandaments per sobre són secundaris i afecten després a no ser que siguin incompatibles. A l'exemple 2.2 com a la primera línia primer apliquem cursiva i després per sobre apliquem el text en majúscules només obtenim la lletra en cursiva. A la segona línia la situació és a la inversa i només obtenim la lletra en majúscules petites. Veiem també com un comandament local com és el canvi de mida no és incompatible amb els altres dos, encara funciona tot i haver-hi un error de sintaxi.

2.2 Justificació i espaiat del text

De forma similar amb la font de la lletra, podem modificar la justificació i espaiat del text amb comandaments amb argument o amb comandaments locals. També veurem un nou tipus de comandament que és de tipus local, només que el seu efecte és encara més localitzat ja que només afecta al lloc on hem posicionat el comandament al codi.

Si no apliquem cap entorn o configuració extra al preàmbul el text pla normalment està justificat a tota l'amplada de línia[2]. Com hem vist a l'exemple 1.7 podem canviar aquesta justificació amb entorns com `flushright` o `center`. El primer justifica tot el text al marge dret, el segon justifica el text al centre. Anàlogament a `flushright` podem fer servir `flushleft` per a justificar el text al marge esquerra (aquesta és la configuració per defecte en editors de text com Word). Podem fer servir aquests entorns de forma localitzada amb els seus anàlegs de comandaments locals, respectivament, `\flushright`, `\centering`, `\flushleft`. Escollir si hem de treballar amb els comandaments d'entorn o locals depen tant del nostre flux de treball com de la compatibilitat amb altres comandaments que estiguem utilitzant. Com a petita guia podeu fer servir el següent flux de treball: fem servir comandaments d'entorn per a delimitar paràgrafs de text, però, per a treballar amb objectes que no siguin text fem servir comandaments locals sota els entorns adequats.

A part de poder configurar la justificació del text també podem canviar algunes distàncies concretes deixant espais en blanc amb una mesura precisa. Això ho aconseguim amb comandaments com `\hspace{...}` o `\vspace{...}`. Aquest comandaments, com hem dit abans, funcionen de forma local i justament a la part del codi on els col·loquem, els podem anomenar comandaments doblement locals. El primer genera un espai en blanc horitzontal, el segon fa el mateix però en vertical. L'argument de cada un d'aquests comandaments accepta la distància que ocuparà en el document aquest espai en blanc. Aquesta distància es pot posar amb diverses unitats, ja siguin centímetres, mil·límetres, punts, etc. S'ha de tenir cura, però, ja que un mal ús de qualsevol dels dos comandaments ens pot fer obtenir resultats no desitjats.

Exemple 2.3

```
\documentclass{article}

\begin{document}
Aquest text es troba justificat de forma usual, sense estar dins de cap
entorn. \hspace{2cm} Aquesta altra part tambe te una justificacio normal
 pero li afecta un espai en blanc de 2 cm.\\ \vspace{1cm}

Aquest salt de linia hauria d'apareixer amb un salt de linia igual al
del text, pero, es veu afectat per l'espai en blanc d'1 cm.\\\\
Tot i que no sigui d'us comu, tambe es poden afegir arguments amb
distancia negativa, es a dir apropant els objectes que venene separats
pel comandament que fem servir:

\begin{center}
    Text exemple a l'esquerra \hspace{-0.5cm} Text exemple a la dreta
\end{center}
\end{document}
```

Sortida:

Aquest text es troba justificat de forma usual, sense estar dins de cap entorn. Aquesta altra part tambe te una justificacio normal pero li afecta un espai en blanc de 2 cm.

Aquest salt de linia hauria d'apareixer amb un salt de linia lleugerament mes gran que l'espai d'interliniat, pero, es veu afectat per l'espai en blanc d'1 cm i apareix mes gran de lo normal.

Tot i que no sigui d'us comu, també es poden afegir arguments amb distancia negativa, es a dir apropant els objectes que venene separats pel comandament que fem servir:

Text exemple a l'esquerraText exemple a la dreta

Observem en aquest exemple 2.1 com aquest espaiat pot oferir-nos molt de joc en cas de voler col·locar text o objectes allà on vulguem. Veurem més endavant (capítol 3) com existeixen més formes d'aplicar

[2]Notem que és a l'amplada de línia i no l'amplada dels marges definit al preàmbul. Aquest detall no és rellevant si treballem a una columna, però veurem que si que s'ha de tenir en compte quan treballem amb més d'una columna (veure secció 2.4)

espaiat horitzontal amb mesures ja predefinides per la mida del text.

Un primer exemple de resultats no desitjats que hem mencionat abans és que amb l'espaiat horitzontal pot passar que una part del text o l'objecte que hi hagi a prop se surti dels marges o de la pàgina del document, quedant tallats i amb un error de sintaxi a LaTeX[3]. Un segon exemple, ara amb l'espaiat vertical, és que l'entorn on intentem cridar aquest espai en blanc sigui o bé un entorn directament incompatible amb aquest comandament o bé que aplicant-ho trenquem l'estructura lògica (o sintaxi) de la resta de paràgrafs. En aquests cas LaTeX interpreta que és un error i compila el document ignorant que existeix aquest espaiat. En cas de tenir algun problema que vingui d'aquest comportament dels comandaments d'espaiat ja s'ha de mirar detingudament si existeix alguna alternativa per a poder aplicar l'espai que estem buscant. Depenent de l'entorn on treballem la solució pot ser forçar a LaTeX a ignorar la sintaxi (això és una opció en alguns entorns), fer servir paquets que ofereixen noves formes d'aplicar espaiats, etc. Com a darrer exemple podem fixar-nos novament en l'exemple 2.1, on apliquem un espaiat horitzontal negatiu i les lletres del text s'encabalquen.

Com a darrer comandament d'espaiat basic tenim un altre de local que serveix per a saltar a una pàgina en blanc nova des d'on se situa el comandament. Per a cridar-ho només es necessita escriure `\newpage` per a poder indicar a LaTeX on tallar el darrer paràgraf i continuar la resta del text a una pàgina següent. Aquest comandament també pot ser sensible en quant a compatibilitat amb alguns entorns provinents de paquets.

2.3 Llistes

Amb les eines que tenim ja podriem construir llistes similars a les que podem escriure amb altres editors de text. Per a fer-ho veiem el següent exemple on aprofitem la indentació natural de la separació de paràgrafs per a poder separar del marge esquerra cada ítem de la llista.

Exemple 2.4

```
\documentclass{article}

\begin{document}
\begin{center}
    \textsc{La meva primera llista}
\end{center}
Ara posarem un exemple d'una llista amb quatre items:\\

- Primer item\\

- Segon item\\

- Tercer item. Aquest tercer item el farem mes llarg que la resta per a
observar que es el que pasa si intentem fer una llista nomes aplicant
els coneixements que tenim fins ara.\\

- Quart item
\end{document}
```

Sortida:

[3] Aquest error no és crític, el compilador de LaTeX només ens avisa de que és un possible error i ens aconsella corregir-lo, però, no deixarà de compilar el que nosaltres li estiguem enviant.

La meva primera llista

Ara posarem un exemple d'una llista amb quatre items:

- Primer item

- Segon item

- Tercer item. Aquest tercer item el farem mes llarg que la resta per a observar que es el que pasa si intentem fer una llista nomes aplicant els coneixements que tenim fins ara.

- Quart item

Veiem com a l'exemple 2.4 si intentem afegir un ítem amb una mida més gran que l'amplada de línia automàticament salta sense indentació. Tot i que això es pot corregir aplicant els comandaments que coneixem hem de dir que és una correcció una mica enrevesada i pot fer farragosa la tasca de crear qualsevol llistat. És per aquest motiu que LaTeX inclou un entorn ja predefinit per a poder generar llistats i poder personalitzar-los lleugerament. Aquest entorn ve delimitat pel comandament `itemize`, posem un exemple de com es fa servir:

Exemple 2.5

```
\documentclass{article}

\begin{document}
A continuacio posarem un exemple de l'entorn \texttt{itemize} amb alguna
 configuracio canviada.
\begin{itemize}
    \item Primer item
    \item[$\star$] Segon item
    \item[$\square$] Tercer item. Fem tambe aquest item mes llarg per a
    comprovar que la indentacio queda com nosaltres volem sense haver de
     configurar cap cosa mes.
    \item[-] Quart item
\end{document}
```

Sortida:

A continuacio posarem un exemple de l'entorn `itemize` amb alguna configuracio canviada.

- Primer item

⋆ Segon item

□ Tercer item. Fem tambe aquest item mes llarg per a comprovar que la indentacio queda com nosaltres volem sense haver de configurar cap cosa mes.

- Quart item

Aquest exemple 2.5 deixa veure clarament com l'entorn `itemize` deixa totes les seves entrades amb una correcta indentació, a més que permet modificar el simbol de l'ítem (el marcador) segons volguem aplicant els comandaments corresponents. De moment no cal que us preocupeu pels símbols ni la seva sintaxi ja que en parlarem d'ells al Capítol 3. Com a darrer exemple d'aquest apartat us mostrarem que també es poden fer llistes anidades, permetent-nos així aplicar la jerarquía que volguem a les nostres llistes.

Exemple 2.6

```
\documentclass{article}

\begin{document}
\begin{itemize}
    \item Primer item
    \item Segon item
```

```
7         \begin{itemize}
8             \item Tercer item, aquest item es troba dins d'un altre \texttt{
              itemize}, per tant es troba en un nivell inferior i LaTeX li
              assigna un altre marcador i un grau d'indentacio mes.
9             \begin{itemize}
10                \item Quart item, aquest es troba a un nivell mes baix
                  encara
11                \item Cinque item
12            \end{itemize}
13        \end{itemize}
14        \item Sise item
15    \end{itemize}
16    \end{document}
```

Sortida:

- Primer item
- Segon item
  - Tercer item, aquest item es troba dins d'un altre `itemize`, per tant es troba en un nivell inferior i LaTeX li assigna un altre marcador i un grau d'indentacio mes.
    - Quart item, aquest es troba a un nivell mes baix encara
    - Cinque item
- Sise item

Notem com de forma automàtica LaTeX aplica un o altre marcador segons el nivell de l'ítem dins de la jerarquia. Amb comandaments més avançats es poden configurar quins són aquests marcadors per defecte per aquells que nosaltres volguem.

2.4 Múltiples columnes

Per defecte LaTeX als seus tipus de documents més usuals aplica un format de text d'una sola columna. Amb aquest format ja ens és suficient per a poder treballar amb totes les eines que ens ofereix LaTeX, però existeixen comandaments que ens permeten treballar a dos o més columnes. Aquesta configuració és una mica delicada, en tenir com a condició que el compilador ha d'entendre el codi que escribim doncs hem d'adaptar algunes eines comunes al format de múltiple columna. Per exemple, els entorns per a col·locar figures o taules no funcionen correctament dins d'un entorn amb múltiples columnes. Tots aquests problemes acaben tenint una solució, més o menys complicada. En aquest apartat ens centrarem només en donar certs exemples de com cridar aquests entorns tant al preàmbul com en mig del document. A la resta del curs farem comentaris de si alguna funcionalitat en concret funciona correctament amb doble columna o no i alguna solució rellevant.

No és lo més usual, però com hem dit podem aplicar comandaments que afecten a tot el document ficant-los al preàmbul. En cas de voler aplicar doble columna a tot el document des del principi, podem posar a la configuració del tipus de document el paràmetre `twocolumn`. LaTeX entén per defecte que volem el text distribuït en dos columnes. A més també entén que aquesta distribució s'ha de completar per a omplir tot l'espai en blanc possible de l'amplada de línia. Si escribim a doble columna una quantitat de text que hauria d'omplir una columna sencera (és a dir, la meitat esquerra del document) només amb aquest comandament no sortirà el que s'esperaria. LaTeX no acaba la primera columna i, al arribar al límit inferior de la pàgina, salta a la següent columna. El que fa per defecte és directament omplir totes dues columnes alhora i de forma coherent. En el cas de tenir aquesta quantitat de text que hem dit abans, LaTeX el ditribuïria en dues columnes que arribarien d'alçada fins a la meitat del document.

Per a treballar a doble columna (o amb més de dues, però això és més inusual) podem escriure dins de l'entorn `multicols`, del paquet `multicol`. Abans de poder fer-lo servir asegureu-vos que teniu el paquet declarat al preàmbul. Per a poder posar-vos exemples amb text suficient per a que es puguin

apreciar les distribucions de columnes farem servir el clàssic *Lorem Ipsum*.[4]

Exemple 2.7

```
\documentclass{article}

\begin{document}
Fora de l'entorn multicols el text es distribueix en una sola columna,
com es habitual. El que ve ara es troba distribuit en dues columnes:\\

\begin{multicols}{2}
    Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do
    eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim
    ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut
    aliquip ex ea commodo consequat. Duis aute irure dolor in
    reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla
    pariatur. Excepteur sint occaecat cupidatat non proident, sunt in
    culpa qui officia deserunt mollit anim id est laborum.
\end{multicols}
\vspace{4mm}
\noindent Tambe podem aplicar un format de tres columnes:\\

\begin{multicols}{3}
    Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do
    eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim
    ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut
    aliquip ex ea commodo consequat. Duis aute irure dolor in
    reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla
    pariatur. Excepteur sint occaecat cupidatat non proident, sunt in
    culpa qui officia deserunt mollit anim id est laborum.
\end{multicols}
\end{document}
```

Sortida:

Fora de l'entorn multicols el text es distribueix en una sola columna, com es habitual. El que ve ara es troba distribuit en dues columnes:

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Tambe podem aplicar un format de tres columnes:

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

De l'exemple 2.7 podem veure que per a indicar el nombre de columnes que volem a la nostra distribució del text només cal que posem el nombre a l'argument de l'entorn. No es pot fer explícitament en un exemple com l'anterior ja que dins d'una tcolorbox[5] no té sentit, però `multicols` accepta forçar la distribució del text a omplir primer la primera columna i, quan s'acabi l'espai, començar a omplir la segona (la distribució de text que potser esperavem en un principi, com hem comentat abans). Això s'indica escrivint `\begin{multicols*}{...}`. L'asterisc al declarar un entorn normalment significa ignorar

[4] El *Lorem Ipsum* és un text d'exemple en llatí que no té significat. Es fa servir per a ocupar espais on hauria d'haver text amb infomació, és útil per a veure com s'acabaria veient algun format de text o com quedarien plantilles.

[5] Veure secció 6.1.1

alguna configuració específica de l'entorn, en aquest cas fa ignorar la redistribució del text en les columnes.

Naturalment la justificació i la configuració del format del text és compatible i s'aplica de la mateixa forma que amb el text distribuït en una sola columna. Com s'ha comentat abans, la justificació del text es fa respecte a l'amplada de línia, per tant si apliquem una justificació cap a la dreta, el text es distribueix en les columnes que haguem indicat i justificat al marge de columna dret més proper que trobi.

2.5 Estructura per a un document científic

En la darrera secció d'aquest capítol farem servir totes les eines que hem aprés i aprendrem de noves amb l'objectiu de poder redactar el nostre primer document. Com la resta del curs aquestes eines i exemples són enfocades a un àmbit científic, però totes es poden fer servir en altres contexts molt diferents.

2.5.1 Pàgina de títol

En diverses ocasions ens veurem amb la necessitat de presentar un treball amb una portada i amb un índex on puguem veure el llistat de continguts del document. Jugant amb la justificació, l'espaiat i el format del text podem crear una pàgina inicial que servirà com a portada, la pàgina de títol. Posant el format del text amb negreta, canviant la mida i manualment posant números podem separar els diferents apartats en seccions. Normalment en un text científic separem la informació en cinc parts: resum o *abstract*, introducció i metodologia, resultats i discussió, conclusions, i bibliografia. Posem-ne primer un exemple de títol generat automàticament per LaTeX:

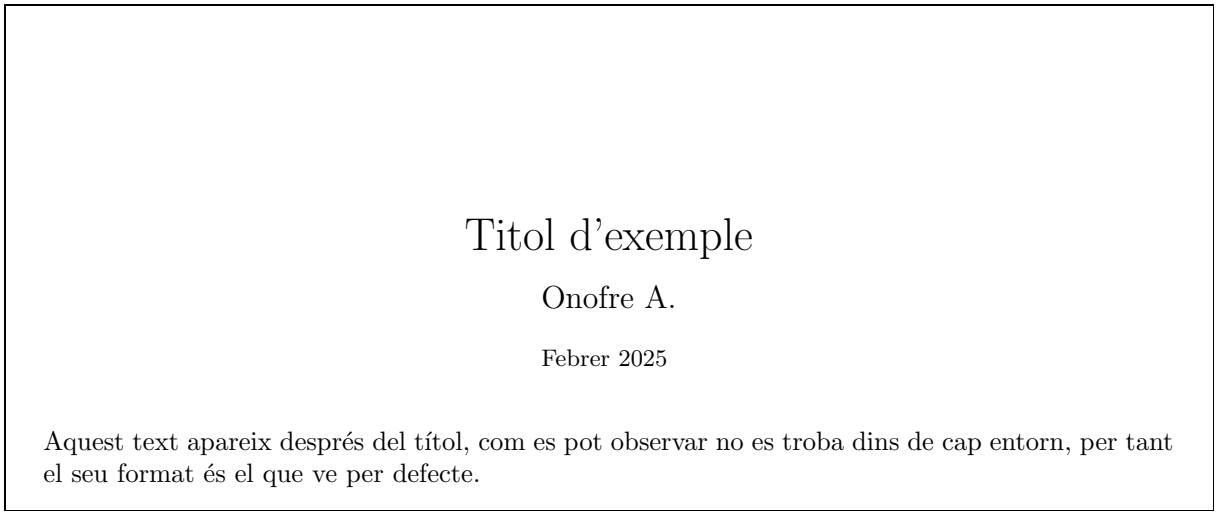
Exemple 2.8

```
\docmentclass{article}

\title{Titol d'exemple}
\author{Onofre A.}
\date{Febrer 2025}

\begin{document}
\maketitle
Aquest text apareix despres del titol, com es pot observar no es troba
dins de cap entorn, per tant el seu format es el que ve per defecte.
\end{document}
```

Sortida:



Veiem a l'exemple 2.8 que existeix el comandament `maketitle`. Aquest comandament genera de forma automàtica un títol amb l'autor i la data agafant la informació que afegim al preàmbul amb els comandaments `title`, `author` i `date`. Overleaf per defecte genera un document en blanc amb aquest títol agafant la informació que afegim al crear-lo. Com veiem el comandament genera un espaiat amb el marge superior, justifica al centre i separa la informació amb un salt de línia més gran de lo normal.

Ens podem quedar amb aquest títol per defecte o crear els nostres propis títols amb les eines que us hem presentat abans.

En cas de voler separar el títol en una pàgina apartada de la resta del text, podem fer servir altres eines d'espaiat del text per a generar alló que realment volem o també un entorn específic per a generar una portada. Començem amb l'entorn. Podem indicar a LaTeX que el text personalitzat dins de l'entorn `titlepage`. Aquest entorn aïlla el text que posem a dins a la primera pàgina, fent que la resta del text vingui a continuació a partir de la segona pàgina i deixant una "marca" que fa que el compilador entengui que allò que hi ha a dins és una portada amb un títol[6]. Podem posar un exemple senzill de títol:

Exemple 2.9

```
\documentclass{article}

\begin{document}
\begin{titlepage}
    \centering
    \vfill
    {\LARGE \textsc{Titol personalitzat d'exemple}}\\
    \vspace{4mm}
    {\large \textbf{Autor:} Onofre A.}\\
    {\large Optica't UAB}
    \vfill
\end{titlepage}
Aquest text d'exemple comena a una nova pagina en blanc.
\end{document}
```

Sortida:

TITOL PERSONALITZAT D'EXEMPLE

Autor: Onofre A.
Optica't UAB

Aquest text d'exemple comença a una nova pagina en blanc.

LaTeX pot fer aquesta tasca de separar la informació al nostre gust de forma pràcticament automàtica segons el propòsit d'questa informació dins del text (com veurem a la secció 2.5.2). A més també és capaç de generar automàticament un índex ordenat i polit sense haver d'escriure res de forma explícita.

2.5.2 Capítols, seccions, subseccions i índex

Com hem dit a la secció anterior, podem separar el tipus d'informació segons el seu context o propòsit. Una de les formes més utilitzades per a separar la informació és l'ús dels comandaments que separen i jerarquitzen la informació. Us deixem una llista de tots els nivells en que es pot separar la informació, ordenats de major "importància" a menor dins del text:

- `part` - Pensat per a separar la informació en blocs molt grans, com per exemple per a separar un text en un bloc de teoria i un bloc d'exercicis. Les parts s'enumeren amb numeros romans, i el títol de cada part surt separat en una nova pàgina en blanc. Aquest comandament es pot fer servir en tipus de document com `book`, `report` o `article`.

[6] Aquesta informació extra per al compilador sembla irellevant però pot ser útil per a algunes automatitzacions com per exemple per a modificar de forma més senzilla com es contabilitzen les pàgines del document. Si fem entendre a LaTeX que existeix una portada, podem fer que entengui que aquella pàgina no s'ha d'indexar amb un nombre de pàgina.

- `chapter` - Separa la informació en blocs grans, per capítols. És posible configurar-ho per a que el títol del capítol estigui en una pàgina separada o que seguidament comenci el text del document. Cada capítol ve enumerat per un nombre natural, començant per l'u. Aquest comandament només es pot utilitzar en aquells tipus de document com `book` i `report`.

- `section` - Separa el text en seccions que, per exemple, poden separar el cos d'un capítol. En textos científics normalment serveix per a separar el document en les 5 parts que hem mencionat anteriorment. Quan una secció es troba en un tipus de document que no se separa per capítols (com en un `article`) la secció s'enumera amb un nombre natural, començant per l'u. Quan una secció es troba dins d'un codument que se separa per capítols, l'enumeració de la secció s'estrucura com $n.m$, on n és el nombre del capítol on es troba la secció i m l'enumeració de la secció dins del capítol. Per exemple, la secció 3.7 d'un document és aquella que es troba al tercer capítol del document i ocupa la setena posició dins de totes les seccions que hi ha dins d'aquell capítol.

- `subsection` - Seguint la lògica que hem vist a `section`, aquesta separació d'un nivel més baix es comporta exactament igual però s'enumera amb el format $m.k$ si es troba en un document sense capítols i amb el format $n.m.k$ si es troba en un document separat per capítols. Anàlogament al punt anterior, cada índex correspon a un nivell de separació, el més alt es troba a l'esquerra i va baixant anant cap a la dreta. Per exemple, la subsecció 1.5.2 pertany al primer capítol, secció cinc i la segona de les subseccions que conté aquesta secció. Un altre exemple, si no hi ha capítols, la subsecció 2.3 pertany a la segona secció del document i ocupa la tercera posició de totes les subseccions que conté la secció on es troba.

- `subsubsection` - Aquesta darrera separació es comporta igual que les dues anteriors només que en un nivell més baix. La seva enumeració depèn de la configuració del document. Si treballem amb documents com `article` la subsubsecció apareixerà enumerada amb el format $m.k.j$ (secció, subsecció i subsubsecció). Si treballem amb documents que permeten separar per capítols normalment les subsubsecions apareixen sense enumerar. El motiu d'això és la configuració per defecte d'aquest tipus de document, amb allò que podem anomenar "profunditat de nivell". Si es vol podem canviar això i fer que les subsubseccions apareixin enumerades amb el format $n.m.k.j$ (capítol, secció, subsecció i subsubsecció) només cal canviar la configuració dins del preàmbul afegint el comandament `\setcounter{secnumdepth}{3}`. El 3 indica el nivell màxim al que volem que aparegui enumerat, en aquest cas `subsubsection` es troba al tercer nivell. Les anteriors separacions també tenen enumerat el seu nivell, -1 (part), 0 (capítol), 1 (secció) i 2 (subsecció). Notem com la profunditat de nivell d'un article és per defecte 3 mentre que per a un book és 2

- `paragraph` - Els nivells que queden normalment no s'utilitzen ja que amb les separacions de línia que hem explicat abans ja és suficient. Igualment aquesta separació correspon al nivell 4, per tant ve sense enumerar per defecte. Una diferència amb la resta de separacions és que no separa el text que fiquem a l'argument del comandament amb la resta del text, només el posa en negreta. A la pràctica el funcionament d'aquesta separació és com posar el text en negreta.

- `subparagraph` - Aquesta és la darrera separació bàsica d'un text en LaTeX, es comporta pràcticament com un text normal en negreta. Es diferencia del paràgraf en moments concrets com per exemple al col·locarse després d'un objecte que no és un paràgraf, per exemple una imatge o un llistat. Un paràgraf després d'un objecte apareix en negreta i no pateix indentació inicial, un subparàgraf si que se li aplica aquesta indentació. El nivell del subparàgraf, naturalment, és el 5.

LaTeX entén que l'enumeració de cada separació és la que correspon a l'ordre d'aparició en el codi. Per exemple, podem tenir dues seccions conscutives anomenades "Secció A" i "Secció B". En un primer moment escrivim la "Secció A" i després la "Secció B" i quan compilem apareixen enumerades amb un 1 per a la A i un 2 per a la B. Si temps després volem canviar d'ordre, és a dir, que apareixi primer al text la "Secció B" i després la "Secció A" podem moure dins del codi cada secció sense cap problema. Ara bé, ara l'enumeració s'haurà actualitzat, ara amb el canvi d'ordre les seccions s'enumeren amb un 1 per a la B i un 2 per a la A. A més, llevat dels nivells -1 (part) i 0 (capítol)[7], totes les separacions són compatibles amb el format de dos (o més) columnes.

[7] Aquests nivells són incompatibles amb l'entorn `multicols`, però si les múltiples columnes estàn configurades al preàmbul i afecten a tot el text si que és pot fer servir `part` i `chapter`. Les incompatibilitats apareixen, per exemple, al intentar ficar un `chapter` dins d'un `multicols`.

Una altra característica de les separacions del text és que els nivells del -1 (part) al 2 (subsection) l'argument que escrivim es troba separat del text normal en forma de títol amb una mida de lletra diferent. Depèn del tipus de document que escollim, però la mida de cada separació normalment és: -1 (part) `huge`, 0 (capítol) `LARGE`, 1 (secció) `Large`, 2 (subsecció) `large` i la resta de nivells `normalsize`, sense separar-se com a un títol.

Aquestes separacions que acabem de descriure són molt útils per a poder-nos moure pel document de forma ràpida, fins i tot si és un document molt llarg. Overleaf té una part de la seva pantalla de treball dedicada a l'ordre i mobilitat dins del document. En aquesta part podrem observar en forma de llistat desplegable totes les separacions que haguem posat al nostre document. Aquest desplegable ens permet navegar entre capítols, seccions i subseccions de forma ràpida.

Un del avantatges més notables de separar la informació d'aquesta forma és que LaTeX és capaç de crear un índex de forma automàtica amb només un comandament: `\tableofcontents`. Aquest comandament es posa dins del text, situat a la pàgina on vulguem situar l'índex. La posició de l'índex dins del text és indiferent, la informació que emmagatzema Overleaf per a poder generar el navegador que hem mencionat abans és la mateixa que ens crea l'índex. Com exemple d'index generat d'aquesta manera podeu mirar l'índex d'aquestes notes del Curs de LaTeX, només que nosaltres hem ficat una característica més que explicarem més endavant, el nostre índex permet navegar pel mateix pdf seleccionant la part a on es vol anar (veure secció 6.1.4).

2.5.3 Footnotes i etiquetes

Com segurament haureu vist ja diverses vegades al llarg d'aquestes notes de tant en tant fiquem aclaracions al text que estem redactant[8]. Es tracta de notes a peu de pàgina o *footnotes* i aplicat a text normal és molt fàcil de fer servir. Només cal fer escriure el comandament `\footnote{...}`, que és un comandament doblement local, és a dir, que actua justament al lloc on el posam al codi. L'argument del comandament és directament el text que sortirà a peu de pàgina i, amb algunes limitacions, es pot emplenar com si estiguessim escrivint al document de forma normal. Podem fins i tot afegir objectes matemàtics (veure Capítol 3).

Pel fet de ser un comandament doblement local s'ha de tenir cura d'on situem una *footnote*, ja que és un comandament bastant sensible a incompatibilitats. Per exemple, com veurem més endavant, posar una *footnote* dins d'un títol d'una taula o figura (anomenat *caption*, veure Capítol 4) és una mica més complicat que simplement afegir allà el comandament de nota a peu de pàgina. Ho explicarem amb més detall més endavant al detallar l'us de figures i taules, però fem un petit avenç per a explicar com solucionar aquest problema. Objectes com un *caption* es troben dins d'entorns anomentats entorns flotants. Aquests entorns pateixen unes normes especials a l'hora de posicionar els objectes que afecta dins del document, per questions d'optimització d'espai. El comandament `footnote` és sensible a entorns flotants, és per això que en aquest tipus d'espai dins del codi hem de separar el marcador i el text de la *footnote*. El marcador és el número que indexa a la *footnote*, és l'objecte que funciona de forma doblement local, el número es queda allà on l'hem posicionat dins del text. És després, quan es compila el document, que la informació d'aquest marcador (posició al text, número que li correspon...) es relaciona amb el text que volem que mostri (el cos de la *footnote*). Per a separar el marcador del text fem servir el comandament `\footnotemark`. Més endavant, podem declarar el text que anirà en aquesta *footnote* escrivint el comandament `\footnotetext{...}`. Aquest darrer comandament, com es pot esperar, s'ha d'escriure a fora de l'entorn flotant.

Les notes a peu de pàgina es poden configurar de diverses maneres, fins i tot utilitzant paquets. Però, LaTeX té algunes configuracions senzilles per defecte. Es pot observar al llarg d'aquestes notes com l'índex de les *footnotes* es reinicia a cada capítol que passa. Aquest comportament és per defecte i ajuda a no acumular un nombre molt gran de notes. Podem manipular l'índex que mostra el marcador, no només amb números, sinò també fent servir símbols.

Per a reiniciar el comptador dels índex que estem fent servir (números, lletres o símbols) només cal escriure allà on vulguem que passi el comandament

```
\setcounter{footnote}{0}
```

[8] Com per exemple aquesta aclaració que esteu llegint ara.

Aquest tipus de comandament és un de nou, s'encarrega de canviar la configuració d'una característica de LaTeX fora del preàmbul. Podem anomenar a aquest tipus de comandaments com a comandaments de configuració. Notem com aquest comandament té doble argument. `setcounter` en el seu primer argument selecciona el llistat d'índex que actualment fa servir l'objecte `footnote`, en el seu segon argument declara quina posició de la llista d'index es vol seleccionar a partir d'aquell moment. En aquest cas el que fa és reiniciar el comptador de números a la posició 0 (o posició 1, és indiferent) de la llista de números naturals que fa servir LaTeX. La pròxima *footnote* que aparegui al text apareixerà amb un 1 com a marcador.

Per a poder fer servir marcadors que no siguin números hem de fer servir un altre comandament de configuració:

```
\renewcommand{\thefootnote}{\fnsymbol{footnote}}
```

per a que els marcadors siguin símbols,

```
\renewcommand{\thefootnote}{\alph{footnote}}
```

per a que siguin lletres minúscules,

```
\renewcommand{\thefootnote}{\roman{footnote}}
```

per a que siguin nombres romans.

Els comandaments de configuració del tipus `renewcommand` normalment el que fan és substituir alguna configuració per defecte de LaTeX o generar una nova ordre. En aquest cas el que fa és substituir la llista de marcadors que hi ha per defecte, passem de la llista de números naturals a una llista amb l'abecedari, una altra amb nombres romans i una altra amb 9 símbols diferents. S'ha de tenir en cura de que, per a llistes finites com l'abecedari o els símbols no fiquem més *footnotes* que elements hi hagi a la llista de marcadors. En cas d'haver posat 9 notes fent servir el llistat de 9 símbols com a marcadors, haurem de reiniciar manualment el comptador o començar a fer servir altres marcadors per a les següents *footnotes*.

Haver de preocupar-se d'un nombre limitat de *footnotes* pot ser una mica farragós. Per aquest motiu existeixen eines externes a LaTeX bàsic en forma de paquet que ens poden ser de molta utilitat. Us deixem un exemple d'un d'aquests paquets i una possible configuració útil. En cas de voler fer servir com a marcadors la llista de símbols i voler configurar el document per a que a cada nova pàgina el comptador de marcadors torni a zero, podem utilitzar el paquet `footmisc` amb la següent configuració al preàmbul:

```
\usepackage[perpage, symbol]{footmisc}
\renewcommand{\thefootnote}{\fnsymbol{footnote}}
```

Podem notar com el primer comandament crida un nou paquet amb una certa configuració (aquella que permet reiniciar el comptador per cada pàgina i aquell que ho fa compatible amb els símbols). El segon comandament canvia la configuració del document com hem vist abans.

Per últim, també podem forçar a que aparegui un marcador concret a una footnote específica. Només cal aplicar un segon argument al comandament `footnote`, indicant la posició a la llista de marcadors d'aquell que vulguem fer servir. Per exemple, si vull fer servir la lletra b com a marcador de la primera *footnote* del meu text, he d'escriure primer el comandament que canvia la llista per defecte a la llista de lletres i després fer servir, allà on toqui, el comandament: `\footnote[2]{Text exemple}`.

Capítol 3

L'entorn matemàtic

3.1 Formes de cridar un entorn matemàtic

3.2 Sintaxi de l'entorn matemàtic

3.2.1 Operacions bàsiques

3.2.2 Parèntesis i claudàtors

3.2.3 Alguns accents, simbols i formats útils

3.2.4 Límits, sumatoris, productoris i integrals

3.3 Matrius i alineació

3.4 Exemples

Capítol 4

Figures i taules

4.1 Figura

4.2 Subfigura

4.3 Taula

Capítol 5

La bibliografia

5.1 L'arxiu `.bib`

5.2 Referenciar i imprimir la bibliografia

Capítol 6

Personalització avançada

6.1 Paquets útils

6.1.1 colorbox

6.1.2 babel

6.1.3 fancyhdr

6.1.4 hyperref

6.1.5 wrapfig

6.1.6 mhchem