زبان Go با استفاده از متغیرهای محیطی GOMAXPROCS در کنار همزمانی از موازی‌سازی (parallelism) هم استفاده می‌شود. هرچند بطور پیش فرض برنامه‌ای که با زبان گو نوشته می‌شود از تمامی هسته‌های CPU استفاده می‌کند ولی شما می‌توانید با **GOMAXPROCS** تعداد هسته‌ها را محدود کنید. لازم به ذکر است که در زبان Go شما به طور مستقیم نمی‌توانید اجرای موازی زیربرنامه‌ای را به کامپایلر دیکته کنید و تصمیم‌گیری در این مورد به عهده Go Runtime Scheduler است.

=====================================================================================

تابع main را وقتی می توانید ایجاد کنید که نام پکیج شما main و گوروتین اصلی شما main است. همه گوروتین ها از تابع main شروع می شوند و گوروتین ها بطور همزمان باز می توانند سایر گوروتین ها را اجرا کنند.

زمانیکه شما تابع main را فراخوانی می کنید بخش اصلی و شروع برنامه شما است. و اگر تابع main شما به هر دلیلی متوقف شود یا اتمام شود سایر گوروتین ها از بین می روند.

گوروتین ها چیزی به نام parent یا child ندارند. زمانیکه شما یک گوروتین را اجرا می کنید این گوروتین در کنار سایر گوروتین ها اجرا می شود و کارش را انجام می دهد. زمانی کار یک گوروتین تمام می شود که تابع بازگشت (return) داشته باشد.

زمانیکه یک برنامه گو اجرا می شود. go runtime رشته های (threads) سیستم عامل را راه اندازی می کند که معادل تعداد CPU های logical قابل استفاده برای فرآیند فعلی است. هر یک از logical CPU ها یک هسته مجازی دارد.

=====================================================================================

گوروتین ها را می توان به عنوان یک thread سبک در نظر گرفت که بصورت مستقلانه می توانند همزمان با سایر گوروتین های دیگر کارها را انجام دهند. و همچنین گوروتین ها می توانند به واسطه کانال داده ها را بین هم اشتراک گذاری و منتقل کنند.

---------------------------------------------------------------------------------

کانال یک نوع تایپ است که داده از نوع خاصی را نگه داری میکند و امکان برقراری ارتباط و همگام سازی داده بین گوروتین ها را فراهم می کند. شما می توانید کانال ها را به عنوان خط لوله های در نظر بگیرید که این خط لوله ها به گوروتین ها متصل می شود و باعث برقراری ارتباط بین گوروتین ها می شود. این ارتباط بین گوروتین ها به هیچ قفل صریحی نیاز ندارد (منظورم mutex و …) چون کانال ها بصورت داخلی قفل ها را مدیریت میکند و در زمان های مناسب و مشخص Lock و UnLock می کند.

شما باید این تایپ را برای متغیری تعریف کنید تا بتوانید بواسطه آن متغیر بین گوروتین ها ارتباط برقرار کنیم.

توجه کنید همیشه سعی کنید کانال را با استفاده از تابع **make** ایجاد کنید.

زمانیکه شما یک کانال را به واسطه make ایجاد می کنید در واقع دارید یک instance از ساختار hchan ایجاد می کنید و تمامی فیلدهای این ساختار مقدار پیش فرض میگیرند.

یک نکته بسیار مهمی که وجود دارد در مورد عملیات دریافت باید توجه کنید که یک مقدار خاص از نوع int به داخل کانال فقط یک بار ارسال شده است و فقط یکبار مقدار قابل دریافت از کانال است .

در زبان گو شما می توانید کانال های بافر شده ایجاد کنید. یک کانال بافر دارای مقداری ظرفیت برای نگه داری داده برای کانال بافر است. بزارید کمی ساده تر توضیح دهیم شما برای کانال همانند آرایه و slice می توانید یک ظرفیت مشخصی تعیین کنید که این کانال فقط می تواند این مقدار از داده را نگه داری کند و تا زمانیکه مقادیر از کانال خارج نشود ظرفیت کانال پرخواهد بود و مقدار جدیدی را نمی توانید به کانال ارسال کنید.

مقدار پیش فرض یک کانال nil است و زمانیکه ما یک کانال بدون تابع make تعریف می کنیم مقدار پیش فرضش nil خوهد بود.

یکسری نکات در خصوص کانال nil وجود دارد :

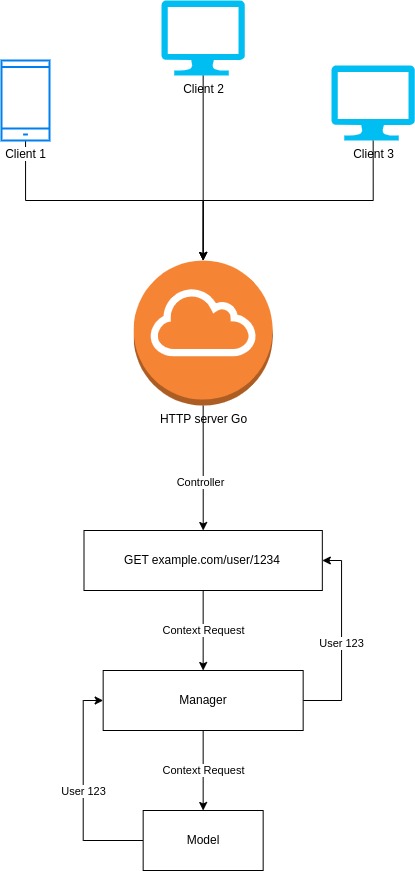
* ارسال داده به یک کانالی که nil است باعث بلاک شدن همیشگی کد شما در آن خط خواهد شد.
* دریافت داده به یک کانالی که nil است باعث بلاک شدن همیشگی کد شما در آن خط خواهد شد.
* بستن یک کانالی که nil باشد باعث panic برنامه شما خواهد شد.

=====================================================================================

در زبان گو select همانند switch می باشد که دارای case و default هستش اما یک فرق کلی دارد که به عملکردش برمیگردد. case های select برای عملیات ارسال و دریافت از کانال منتظر می ماند. در کل شما با استفاده از select می توانید از کانال های مختلف اطلاعات ارسال و دریافت کنید و پس از آن برروی آن اطلاعات عملیات انجام دهید.  
select تا زمانی که یکی از case ها آماده شود بلاک می شود.  
اگر همزمان چندتا case برای انجام عملیات آماده شود select بصورت تصادفی یکی را انتخاب میکند تا عملیات تکمیل شود.

وقتی شما قصد دارید از گوروتین و کانال استفاده کنید در اینجا select نقش خیلی پررنگی در کنترل عملیات کانال ها دارد. اینجاست که بحث همزمانی در زبان گو خیلی زیبا می شود. select می تواند بطور همزمان داده را از کانال دریافت کند و برای اجرا سایر عملیات آماده کند. بنابراین select همراه با کانال و گوروتین خیلی ابزار قدرتمندی می شود برای کنترل و مدیریت همگام سازی و همزمانی.

=====================================================================================



شما فرض کنید یک کلاینت به سرور http راه اندازی کردید که یکسری آدرس API دارد که کلاینت می تواند با استفاده از این آدرس ها با سرور شما ارتباط برقرار کند و یک عملیاتی را انجام دهد. حال وقتی کلاینت درخواست می دهد. درخواست تا زمانیکه کامل شود و خروجی به کاربر نمایش داده شود می توانید این درخواست را بواسطه context در لایه های مختلف پروژه خود منتهی کنی و یکسری عملیات یا اطلاعات را در هر لایه از context بگیرید. اگر به دیاگرام فوق نگاه کنید اگر کلاینت درخواستش را لغو کند و درخواست کاربر به واسط context تا لایه Manager رفته باشد می تواند این درخواست در همان لایه متوقف شود و عملیات تکمیل نشود.

### کاربردهای context

### - لغو یک درخواستی که منتهی شده به لایه های مختلف پروژه بواسطه تابع cancel در پکیج context

### - انتقال داده های حساس به لایه های مختلف بواسطه تابع WithValue در پکیج context

### - گذاشتن timeout برروی context جهت لغو درخواستی که خیلی باعث منتظر ماندن می شود بواسطه تابع WithTimeout در پکیج context

بدنه اصلی یک context از اینترفیس تشکیل شده که یکسری متدها برای مدیریت یک درخواست برروی لایه های مختلف را دارد.

**متد Done :** بواسطه این متد که یک کانال فقط دریافت است شما می توانید سیگنال توقف درخواست را دریافت کنید و خطا برگردانید.

**متد Err :** داخل این متد اینترفیس خطا وجود دارد که خطاهای مربوط به context را می توانید دریافت و مدیریت کنید.

**متد Deadline :** با استفاده از این متد می توانید context هایی که از نوع Deadline هستند را مدیریت کنید.

**متد Value :** با استفاده از این می توانید مقادیری که بصورت key/value داخل context ذخیره شده را دریافت کنید که بصورت اینترفیس یک key میگیرد و به صورت اینترفیس مقدار داخل key را برمیگرداند.

تابع WithDeadline تا حدی شبیه به WithTimeout است اما با این تفاوت که پارامتر زمانی که میگیرد از نوع time.Time است و مدت زمانی که میگیرد براساس تایم هست مثلا شما میگید ۵ ثانیه بعد از زمان الان درخواست را لغو کند در صورتیکه withTimeout مدت زمان میگیرد که درخواست ۵ ثانیه مهلت دارد کارش را اتمام کند.

=====================================================================================

پکیج sync یکی از کاربردی ترین پکیج ها در زمینه همزمانی می باشد و یکسری تایپ و توابع برایهمگام سازی و کنترل دسترسی همزمان به داده مشترک ارائه می دهد.

توجه کنید که پکیج sync فقط و فقط برای مدیریت و همگام سازی دسترسی های گوروتین ها به یک داده مشترک استفاده می شود.

**۲ نکته خیلی مهم**

سعی کنید پس از اینکه تابع Lock را فراخوانی میکنید تابع Unlock را داخل defer قرار دهید.  
زمانیکه قصد دارید Mutex را به عنوان پارامتر ورودی برای توابع تعریف کنید بهتر است از نوع اشاره گر باشد.