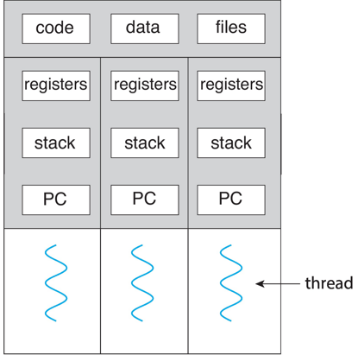
**Threads & Concurrency**

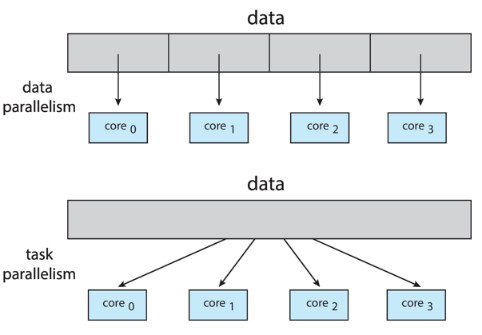
Benifitet e multithreading : *Reagimi* – Lejon që të vazhdoj ekzekutimi i procesit edhe nëse një pjes e procesit është bllokuar. *Ekonomia* – Kushton më pak krijimi i thredave sesa krijimi i proceseve gjithashtu më pak kushton thread switching sesa context switching. *Zgjerushmëria* - Zgjerohen për të ofruar shërbime të reja. *Resouce sharing* – Me qenë se gjenden në të njëjtën hapsir memoruese, pra i përkasin po të njëjtit process, mund të përdorin hapsiren e të dhënave për të komunikuar dhe kjo mënyr është më e lehtë se *message-passing* dhe *shared-memory.*



Një program multithread që të cilsohet si i till edhe duhet të programohet si i till, kjo sjell me vete punët ekstra që duhet bër programeri dhe sfidat me të cilat programeri mund të ballafaqohet, në vijim janë të listuara sfidat me të cilat përballemi gjat krijimit të programeve multithread : *Ndarja e aktiviteteve* – duhet ndarë secilit thread aktivitetin me të cilin do mirret. *Balanci* – Secilit thread duhet dhënë koh e njëjtë e ekzekutimit. *Ndarja e të dhënave*. *Vartësia e të dhënave* – Nuk duhet lejuar që thredat të varen/presin nga njëri tjetri në kryerjen e aktiviteteve në të dhënat e përbashkëta. *Testimi & Debugging* – Është më kompleks.

**Paralelizmi** nënkupton që më shumë se një punë kryhen në të njëjtën kohë.

**Concurrency** nënkupton që më shumë se një punë bëjn progress në ekzekutimin e tyre, mirpo nuk ekzekutohen paralelisht.

**Llojet e paralelizmit** (Ndarjes së punëve) : *Data Paralelism* – Thredat kryejn operacione të njëjta në të dhënat e përbashkëta të cilat i janë ndarë secilit thread. *Task Paralelism* – Threadt kryejn operacione të ndryshme në të dhënat e përbashkëta.

Thredat kur krijohen i kanë dy nivele : *User Thread* – Krijohen dhe menagjohen nga useri duke përdorur thread librariat si *windows thread, java thread, posix pthread*. *Kernel Thread* – Krijohen dhe menagjohen nga sistemi operativ.

Kur krijohen *user threads* bashkëveprimi me *kernel thread* është më se i nevojshëm, për të arritur kët bashkveprim ekzistojnë tri modele : *One-to-Many, One-to-One, Many-to-Many, 2-Level.*

**One-to-Many** : Një *kernel thread* i asocohet shumë *user thredave* përparësi e kti modeli është se na lejon të krijojmë sado *user threada*, mirpo mangësi e ka se nuk e lejon paralelizmin shkaku i një *kernel thredi* të vetëm arsye e njëjtë e cila mund të dërgoj në bllokimin e të gjith *user thredave* nëse ai *kernel thread* bllokohet.

**One-To-One** : Përdoret nga windows dhe linux, ka përparësi në lejimin e paralalizmit dhe largon rrezikun për bllokimin që ndodhte tek m:m, ndërsa mangësi e ka se e kufizon numrin e thredave të cilët mund të krijohen.

**Many-To-Many** : Shumë threda të userit asocohen me shumë threda të kernelit, ka përparsi se nuk e kufizon numrin e thredave mirpo nuk e lejon gjithmon paralelizmin.

**2-Level** : Është model hibrid i cili e lejon edhe Many:Many edhe One:One.